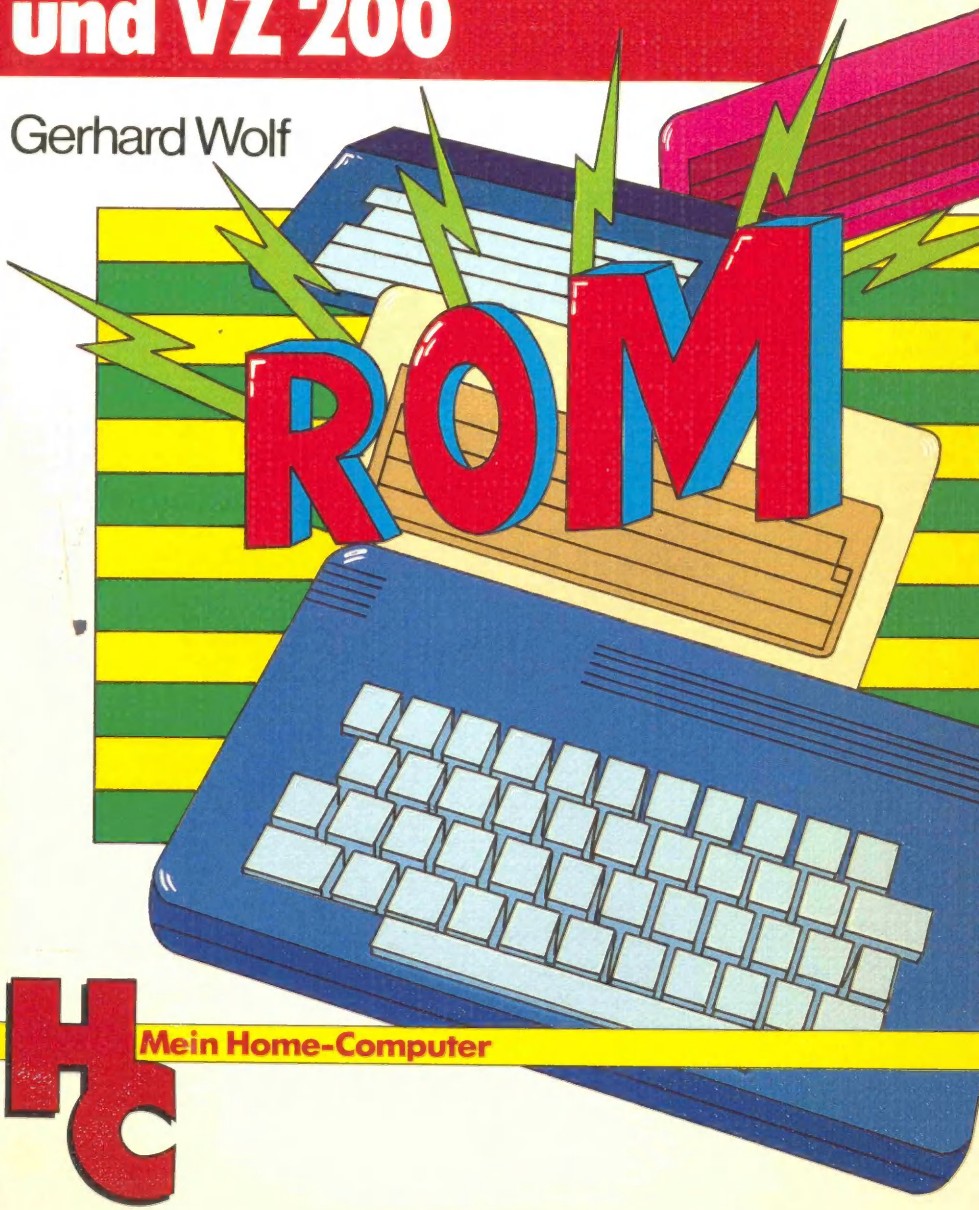
ROM-Listings für Laser 110, 210, 310 und VZ 200

Gerhard Wolf



Mein Home-Computer

Gerhard Wolf

ROM-Listings für Laser 110, 210, 310 und VZ 200

HC – Mein Home-Computer

Gerhard Wolf

ROM-Listings für Laser 110, 210, 310 und VZ 200

Vollständige dokumentierte Auflistung des BASIC-Interpreters Version 2.0

VOGEL-BUCHVERLAG
WÜRZBURG

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Wolf, Gerhard:
ROM-Listings für Laser 110, 210, 310 und VZ 200: vollst. dokumentierte Auflisting d. BASIC-Interpreters Version 2.0 / Gerhard Wolf.
Würzburg: Vogel, 1985.
(HC – Mein Home-Computer)
ISBN 3-8023-0852-2

ISBN 3-8023-0852-2
1. Auflage. 1985
Alle Rechte, auch der Übersetzung, vorbehalten. Kein Teil des Werkes darf in irgendeiner Form (Druck, Fotokopie, Mikrofilm oder einem anderen Verfahren) ohne schriftliche Genehmigung des Verlages reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden. Hiervon sind die in §§ 53, 54 UrhG ausdrücklich genannten Ausnahmefälle nicht berührt.
Printed in Germany
Copyright 1985 by Vogel-Buchverlag Würzburg
Umschlaggestaltung: Bernd Schröder, Böhl
Herstellung: Alois Erdl KG, Trostberg

Das vorliegende Buch enthält eine vollständige, dokumentierte Auflistung des BASIC-Interpreters, Version 2.0 für die LASER - Computer 110, 210 und 310 und den VZ200.

Die Unterschiede zur Version 1.2 beziehen sich ausschließlich auf die Umschaltungsmöglichkeit der Hintergrundfarbe zwischen schwarz und grün im Textmodus. Die dazu erforderlichen Zusatzroutinen sind fast vollständig im ROM-Bereich ab 3E00H untergebracht.
Da im restlichen ROM-Bereich keine Adressverschiebungen vorgenommen wurden, läßt sich die Auflistung bis auf diesen Bereich auch für die Version 1.2 anwenden.

Zunächst einige Vereinbarungen:

Die Zahlendarstellung ist dezimal, wenn nicht eine besondere Kennzeichnung angebracht ist. 'H' hinter einer Zahl kennzeichnet diese als hexadezimal und 'B' als binäre Darstellung.

In der Register-Darstellung sind die allgemeinen Z80-Register A, B, C, D, E, H, L, IX und IY verwendet worden.
In der Arithmetik-Beschreibung werden zusätzlich die Register-Bezeichnungen X und Y angewandt. X ist der ab Adresse 791DH im RAM benutzte Arbeitsbereich für Arithmetik-Operationen jeden Typs. Mit Y sind bei der Arithmetik mit einfacher Genauigkeit die Z80-Register B,C,D,E (B=Exponent, C=MSB, DE=LSB) gemeint, bei der Arithmetik mit doppelter Genauigkeit findet für Y der RAM-Arbeitsbereich ab Adresse 7927H Verwendung.

MSB (most significant byte) = höchstwertiges Byte einer Zahl
LSB (least significant byte) = niederwertiges Byte einer Zahl

Werden einzelne Bits eines Registers oder einer Adresse angesprochen, so ist das entsprechende Bit hinter dem Register oder der Adresse in Klammern angegeben, z.B. A(7) = Bit 7 im Register A, MSB X(0) = Bit 0 im höchstwertigen Byte des Arithmetik-Registers X.

In der Auflistung finden Sie eine Reihe von Byte-Definitionen (DEFB), hinter denen im Kommentar ein vollständiger Befehl mit dem Zusatz "Dummy-Befehl" aufgeführt ist. Diese dienen dazu, bei linearem Durchlauf der entsprechenden Routine, die im Operanden enthaltenen Befehlcodes zu übergehen.

```
                        Initialisierung des Rechners

0000  F3          DI                    ;Interupts ausschalten
0001  AF          XOR   A               ;Text-Modus einschalten
0002  32 00 68    LD    (6800H),A
0005  C3 74 06    JP    0674H           ;weiter bei 674

                        ****************************************

                        Restart 8
0008  C3 00 78    JP    7800H           ;Sprung über RAM-Vektor 7800H
                                        ;zur Adresse 1C96H

000B  E1          POP   HL              ;unbenutzt
000C  E9          JP    (HL)
000D  00 00 00

                        Restart 10
0010  C3 03 78    JP    7803H           ;Sprung über RAM-Vektor 7803H
                                        ;zur Adresse 1D78H

                        Einlesen eines Zeichens über Device-Control-Block (DCB)
0013  C5          PUSH  BC              ;BC retten
0014  06 01       LD    B,1             ;B für DCB-Prüfung setzen
0016  18 2E       JR    46H             ;zur DCB-Aufrufroutine

                        Restart 18
0018  C3 06 78    JP    7806H           ;Sprung über RAM-Vektor 7806H
                                        ;zur Adresse 1C90H

                        Ausgabe eines Zeichens über Device-Control-Block (DCB)
001B  C5          PUSH  BC              ;BC retten
001C  06 02       LD    B,2             ;B für DCB-Prüfung setzen
001E  18 26       JR    46H             ;zur DCB-Aufrufroutine

                        Restart 20
0020  C3 09 78    JP    7809H           ;Sprung über RAM-Vektor 7809H
                                        ;zur Adresse 25D9H

0023  C5          PUSH  BC              ;nicht benutzt
0024  06 04       LD    B,4
0026  18 1E       JR    46H
```

```
                        Restart 28
0028  C3 0C 78          JP      780CH           ;Sprung zum RAM-Vektor 780CH

                        Tastatur-Abfrage
                        Reg. A enthält beim Rücksprung den ASCII-Code einer
                        gedrückten Taste oder 0, wenn keine gedrückt.
002B  11 15 78          LD      DE,7815H        ;DCB-Adresse für Tastatur laden
002E  18 E3             JR      13H             ;weiter bei 13H

                        Restart 30
0030  0F 78             JP      780FH           ;Sprung zum RAM-Vektor 780FH

                        Bildschirmausgabe über DCB
                        beim LASER 110-310 nicht benutzt.
0033  11 1D 78          LD      DE,781DH        ;DCB-Adresse laden
0036  18 E3             JR      1BH             ;weiter bei 1BH

                        Restart 38              ;Interrupt-Vektor für IM1
0038  C3 B8 2E          JP      2EB8H           ;Zur Interrupt-Service-Routine

                        Druckerausgabe über Device-Control-Block (DB)
                        Reg. A muß das auszugebende Zeichen enthalten
003B  11 25 78          LD      DE,7825H        ;DCB-Adresse laden
003E  18 DB             JR      1BH             ;weiter bei 1B

0040  C3 FD 2E          JP      2EFDH           ;zur Tastatur-Leseroutine

0043  C9                RET                     ;nicht benutzt
0044  00 00

0046  C3 74 06          JP      674H            ;Sprung zur DCB-Aufrufroutine

                        Tastaturabfrage
                        wartet, bis eine Taste gedrückt wird.
                        Ausg.: A-Reg enthält ASCII-Code der gedr. Taste
0049  CD 2B 00          CALL    2BH             ;Tastatur auswerten
004C  B7                OR      A               ;Taste gedrückt?
004D  C8                RET     NZ              ;Ja, zurück
004E  18 F9             JR      49H             ;nein, warten

                        ****************************************

                        Zeichen aus Cursor-Position sichern
0050  2A 20 78          LD      HL,7820H        ;Cursor-Adresse laden
```

```
0053  7E              LD      A,(HL)          ;Zeichen laden
0054  32 3C 78        LD      (783CH),A       ;nach 783CH sichern
0057  C9              RET

0058                                          ;nicht benutzt
-
005F

                      ****************************************

                      Zeitschleife
                      Eing.: Reg. BC bestimmt Dauer
0060  0B              DEC     BC              ;Zähler - 1
0061  78              LD      A,B             ;= 0?
0062  B1              OR      C
0063  20 FB           JR      NZ,60H          ;nein, zurück
0065  C9              RET                     ;ja, fertig

                      ****************************************

                      Interrupt-Vektor für 'Non maskable interrupts
                      Im LASER 110-310 nicht benutzt
0066  31 00 06        LD      SP,600H
0069  3A EC 68        LD      A,(68ECH)
006C  3C              INC     A
006D  FE 02           CP      2
006F  D2 00 00        JP      NC,0            ;Kaltstart
0072  C3 CC 06        JP      6CCH            ;Warmstart

                      ****************************************

                      BASIC-Initialisierung Teil 2
0075  11 B0 78        LD      DE,7880H        ;Unterprogramme f. Div, Out, Inp
0078  21 F7 18        LD      HL,18F7H        ;u.a. ins RAM übertragen.
007B  01 27 00        LD      BC,27H
007E  ED B0           LDIR

0080  21 E5 79        LD      HL,79E5H        ;IO-Buffer einrichten
0083  36 3A           LD      (HL),3AH        ;vor Buffer ':' , '0' , ','
0085  23              INC     HL              ;schreiben
0086  70              LD      (HL),B
0087  23              INC     HL
0088  36 2C           LD      (HL),2C         ;','
008A  23              INC     HL
```

```
008B  22 A7 78        LD    (78A7H),HL    ;IO-Bufferadresse speichern

008E  11 2D 01        LD    DE,12DH       ;Vektoren der Disk-Befehle
0091  06 1C           LD    B,28          ;auf 'DISK COMMAND'-Error setzen
0093  21 52 79        LD    HL,7952H      ;Anfangsadr. der Vektoren im RAM
0096  36 C3           LD    (HL),0C3H     ;Sprungbefehl auf Adr. 12DH in
0098  23              INC   HL            ;in jeden Vektor schreiben
0099  73              LD    (HL),E
009A  23              INC   HL
009B  72              LD    (HL),D
009D  10 F7           DJNZ  0096H         ;nächster Vektor

009F  06 15           LD    B,21          ;RAM-Adressen zur Erweiterung
00A1  36 C9           LD    (HL),0C9H     ;bestehender BASIC-Befehle mit
00A3  23              INC   HL            ;RETURN vorbesetzen
00A4  23              INC   HL            ;danach zwei Byte für evtl.
00A5  23              INC   HL            ;Sprungadresse freihalten
00A6  10 F9           DJNZ  00A1H         ;weiter

00A8  21 E8 7A        LD    HL,7AE8H      ;Programmanfang markieren
00AB  70              LD    (HL),B        ;mit 0
00AC  31 F8 79        LD    SP,79F8H      ;Stackpointer laden
00AF  CD BF 1B        CALL  1B8FH         ;Stack initialisieren (über NEW)
00B2  CD C9 01        CALL  1C9H          ;Bildschirm löschen
00B5  00 00 00 00 00                      ;nicht benutzt
00BA  00 00 00 00
00BE  18 04           JR    00C4H         ;nächsten 4 Byte überspringen
00C0  D7 B7 20 12                         ;nicht benutzt

00C4  21 4C 7B        LD    HL,7B4CH      ;Ab 7B4DH Speicherende suchen
00C7  23              INC   HL            ;nächstes Byte
00C8  7C              LD    A,H           ;Adresse 0000 erreicht?
00C9  B5              OR    L
00CA  28 1B           JR    Z,00E7H       ;Ja!
00CC  7E              LD    A,(HL)        ;Inhalt des Bytes laden
00CD  47              LD    B,A           ;merken
00CE  2F              CPL                 ;1er Komplement bilden
00CF  77              LD    (HL),A        ;und speichern
00D0  BE              CP    (HL)          ;ausgelesener Wert gleich?
00D1  70              LD    (HL),B        ;Alten Byteinhalt wiederherst.
00D2  28 F3           JR    Z,00C7H       ;wenn gleich, nächstes Byte
00D4  18 11           JR    00E7H         ;Erm. Speicherende verarbeiten

00D6  CD 5A 1E        CALL  1E5AH         ;nicht benutzt
```

```
00D9  B7              OR      A
00DA  C2 97 19        JP      NZ,1997H
00DD  EB              EX      DE,HL
00DE  2B              DEC     HL
00DF  3E 8F           LD      A,8FH
00E1  46              LD      B,(HL)
00E2  77              LD      (HL),A
00E3  BE              CP      (HL)
00E4  70              LD      (HL),B
00E5  20 CE           JR      NZ,00B5H

                      Speicherende für BASIC festlegen
00E7  2B              DEC     HL              ;Adresse des letzten Bytes
00E8  11 14 7C        LD      DE,7C14H        ;es müssen mindestens 1068 Byte
00EB  DF              RST     18H             ;frei sein
00EC  DA 7A 19        JP      C,197AH         ;sonst "OUT OF MEMORY"-Fehler
00EF  11 CE FF        LD      DE,-50
00F2  22 B1 78        LD      (78B1H),HL      ;Speicherendadresse merken
00F5  19              ADD     HL,DE           ;Speicherende -50
00F6  22 A0 78        LD      (78A0H),HL      ;=Anfang String-Space - 1
00F9  CD 4D 1B        CALL    1B4DH           ;Aufruf der "NEW"-Routine
00FC  CD 84 34        CALL    3484H           ;Zähler u. Pointer initialisieren
00FF  21 0F 01        LD      HL,010FH        ;Adresse Vorstelltext
0102  CD A7 28        CALL    28A7H           ;Text ausgeben
0105  ED 56           IM      1               ;Interrupt-Mode 1 einschalten
0107  C3 8E 06        JP      068EH           ;zum Teil 3 der Initialisierung

                      ************************************

                      Vorstelltext
010F  56 49 44 45     DEFM    'VIDEO TECHNOLOGY'
      4F 20 54 45
      43 48 4E 4F
      4C 4F 47 59
011F  0D              DEFB    0DH
0120  42 41 53 49     DEFM    'BASIC V2.0'
      43 20 56 32
      2E 30
012A  0D 0D 0D        3x DEFB 0DH

                      ************************************

                      Ausgabe "DISK COMMAND - ERROR"
012D  1E 2C           LD      E,2CH           ;Fehlercode
```

```
012F  C3 A2 19        JP      19A2H           ;Sprung zur Meldungsausgabe

                      ************************************
                      Grafikanweisungen POINT, SET und RESET

                      POINT - Anweisung
                      Ermittelt, ob Punkt in hochaufl. Grafik gesetzt ist
0132  D7               RST     10H             ;Nächstes Zeichen d. Befehl
0133  AF               XOR     A               ;0 = Flag für Point
0134  01               DEFB    01H             ;simuliert LD BC,003E bei POINT

                      SET - Anweisung
                      Setzt Punkt in hochauflösender Grafik
0135  3E 80            LD      A,80H           ;80 = Flag für SET
0137  01               DEFB    01H             ;simuliert LD BC,013E

                      RESET - Anweisung
                      Löscht Punkt in hochauflösender Grafik
0138  3E 01            LD      A,01H           ;1 = Flag für RESET

                      Gemeinsam für POINT, SET und RESET
013A  F5               PUSH    AF              ;Flag retten
013B  CF               RST     8               ;nächster Zeichen = '('?
013C  28               DEFB    '('
013D  CD 1C 2B         CALL    2B1CH           ;1. Ausdruck in Klammer auswerten
                                               ;X-Koordinate
0140  FE 80            CP      128             ;> 127?
0142  D2 4A 1E         JP      NC,1E4AH        ;Ja, SYNTAX-ERROR
0145  F5               PUSH    AF              ;X-Koordinate auf Stack
0146  CF               RST     8               ;folgt ein Komma?
0147  2C               DEFB    ','
0148  CD 1C 2B         CALL    2B1CH           ;2. Ausdruck in Klammer auswerten
                                               ;Y-Koordinate
014B  FE 40            CP      64              ;> 63?
014D  D2 4A 1E         JP      NC,1E4AH        ;Ja, SYNTAX-ERROR
                      Aus X,Y Bildadresse und Bitmaske ermitteln
0150  5F               LD      E,A             ;Y-Wert in DE
0151  AF               XOR     A
0152  57               LD      D,A
0153  EB               EX      DE,HL           ;Y * Zeilenlänge (x32)
0154  29               ADD     HL,HL           ;x2
0155  29               ADD     HL,HL           ;x4
0156  29               ADD     HL,HL           ;x8
0157  29               ADD     HL,HL           ;x16
```

```
0158  29           ADD   HL,HL          ;x32
0159  EB           EX    DE,HL          ;= rel. Zeilenanf.adresse
015A  F1           POP   AF             ;X-Wert laden
015B  F5           PUSH  AF             ;und wieder auf Stack
015C  CB 3F        SRL   A              ;X-Koordiante / 4
015E  CB 3F        SRL   A
0160  83           ADD   A,E            ;+ rel. Zeilenanf.adresse
0161  5F           LD    E,A
0162  7A           LD    A,D            ;+ Bildanfangsadresse
0163  F6 70        OR    70H            ;(beim LASER 7000H)
0165  57           LD    D,A            ;DE = Bildadresse
0166  F1           POP   AF             ;X-Koordinate laden
0167  E6 03        AND   3              ;letzte 2 Bits maskieren (0,1,2,3)
0169  87           ADD   A,A            ;x 2 (0,2,4,6)
016A  47           LD    B,A            ;in B als Verschiebezähler

016B  F1           POP   AF             ;Funktions-Flag laden
016C  B7           OR    A              ;= 0?
016D  CA E7 38     JP    Z,38E7H        ;Ja - dann POINT ausführen
0170  F5           PUSH  AF             ;Funktions-Flag sichern
                   Für SET und RESET Bits in Reg. A und C maskieren
0171  0E 3F        LD    C,3FH          ;Grundwert in C laden
0173  3A 46 78     LD    A,(7846H)      ;Farbcode als Grundwert in A
0176  CB 27        SLA   A              ;in obere 2 Bits schieben
0178  CB 27        SLA   A
017A  CB 0F        RRC   A              ;mit B als Schiebezähler Grund-
017C  CB 09        RRC   C              ;werte in erf. Position schieben
017E  10 FA        DJNZ  017AH          ;A für SET (OR), C für RESET (AND)
0180  C3 03 39     JP    3903H          ;weiter bei 3903H

                   ****************************************

                   Fehlerbehandlung bei VERIFY
0183  21 39 78     LD    HL,7839H       ;Verify-Bit in Flag2 löschen
0186  CB 9E        RES   3,(HL)
0188  21 84 03     LD    HL,0384H       ;Fehlermeldung adressieren
018B  CD A7 28     CALL  28A7H          ;und ausgeben
018E  C3 CF 36     JP    36CFH          ;weiter bei 36CF

                   ****************************************

                   INKEY$ - Funktion
019D  D7           RST   10H            ;Nächstes Zeichen
019E  E5           PUSH  HL             ;Pointer retten
```

```
019F  3A 99 78        LD    A,(7899H)      ;Zeichen aus INKEY$-Speicher laden
01A2  B7              OR    A              ;Zeichen vorhanden?
01A3  20 06           JR    NZ,01ABH       ;Ja!
01A5  CD 58 03        CALL  0358H          ;Tastaturabfrage
01A8  B7              OR    A              ;neue Taste gedrückt?
01A9  28 11           JR    Z,01BCH        ;nein, Leerstring in X-Reg.
01AB  F5              PUSH  AF             ;Zeichen auf Stack
01AC  AF              XOR   A              ;INKEY$-Speicher löschen
01AD  32 99 78        LD    (7899H),A
01B0  3C              INC   A              ;Stringlänge = 1
01B1  CD 57 28        CALL  2857H          ;1 Byte im Stringspace reservieren
01B4  F1              POP   AF             ;Zeichen wieder laden
01B5  2A D4 78        LD    HL,(78D4H)     ;Adresse im Stringspace laden
01B8  77              LD    (HL),A         ;Zeichen in Stringspace übertr.
01B9  C3 84 28        JP    2884H          ;weiter bei 2884H

01BC  21 28 19        LD    HL,1928H       ;Zeiger auf Leerstring
01BF  22 21 79        LD    (7921H),HL     ;nach X
01C2  3E 03           LD    A,3            ;Typ = String setzen
01C4  32 AF 78        LD    (78AFH),A
01C7  E1              POP   HL             ;Pointer wieder laden
01C8  C9              RET

                      ****************************************

                      CLS - Anweisung
                      Löschen des Bildschirms
01C9  3E 1C           LD    A,1CH          ;Cursor an Bildanfang
01CB  CD 3A 03        CALL  033AH
01CE  3E 1F           LD    A,1FH          ;Bild bis Bildende löschen
01D0  C3 3A 03        JP    033AH

                      ****************************************

                      RANDOM - Anweisung
                      Initialisieren des Zufallsgenerators
01D3  ED 5F           LD    A,R            ;Refresh-Register laden
01D5  32 AB 78        LD    (78ABH),A      ;in Zufallszahl-Grundwert
01D8  C9              RET

                      ****************************************

                      Tastatur - Tabellen
```

```
                        ****************************************

                        Tasten-Codes ohne SHIFT

01D9  54 47 42 35       DEFB   'T','G','B','5','N','6','Y','H'      Bitreihe 0
      4E 36 59 48
01E1  57 53 58 32       DEFB   'W','S','X','2','.','9','O','L'      Bitreihe 1
      2E 39 4F 4C
01E9  00 00 00 00       DEFB   00H,00H,00H,00H,00H,'-',CR ,':'       Bitreihe 2
      00 2D 0D 3A
01F1  45 44 43 33       DEFB   'E','D','C','3',',','8','I','K'      Bitreihe 3
      2C 38 49 4B
01F9  51 41 5A 31       DEFB   'Q','A','Z','1',' ','0','P',';'      Bitreihe 4
      20 30 50 3B
0201  52 46 56 34       DEFB   'R','F','V','4','M','7','U','J'      Bitreihe 5
      4D 37 55 4A

                        Tastatur-Codes mit SHIFT (u.a. Semigrafik)

0209  8C 89 00 25       DEFB   8CH,89H,00H,25H,5EH,26H,83H,86H       Bitreihe 0
      5E 26 83 86
0211  8D 82 00 22       DEFB   8DH,82H,00H,22H,3EH,29H,5BH,3FH       Bitreihe 1
      3E 29 5B 3F
0219  00 00 00 00       DEFB   00H,00H,00H,00H,00H,3DH,0DH,2AH       Bitreihe 2
      00 3D 0D 2A
0221  8B 84 00 23       DEFB   8BH,84H,00H,23H,3CH,28H,85H,2FH       Bitreihe 3
      3C 28 85 2F
0229  8E 81 00 21       DEFB   8EH,81H,00H,21H,20H,40H,5DH,2BH       Bitreihe 4
      20 40 5D 2B
0231  87 88 00 24       DEFB   87H,88H,00H,24H,5CH,27H,8AH,8FH       Bitreihe 5
      5C 27 8A 8F

                        Tastatur-Codes mit CTRL (u.a. CMD-Token)

0239  CA 8D B5 B4       DEFB   CAH,8DH,B5H,B4H,97H,8EH,95H,84H       Bitreihe 0
      97 8E 95 84
0241  BD CC B1 B9       DEFB   BDH,CCH,B1H,B9H,1BH,8BH,8CH,15H      Bitreihe 1
      1B 8B 8C 15
0249  00 00 00 00       DEFB   00H,00H,00H,00H,00H,01H,00H,00H      Bitreihe 2
      00 01 00 00
0251  87 8A B3 9C       DEFB   87H,8AH,B3H,9CH,09H,BBH,89H,BCH      Bitreihe 3
      09 BB 89 BC
0259  81 9D E5 BA       DEFB   81H,9DH,E5H,BAH,0AH,88H,B2H,7FH      Bitreihe 4
      0A 88 B2 7F
```

```
0261  92 91 AF 98      DEFB   92H,91H,AFH,98H,08H,80H,8FH,93H      Bitreihe 5
      08 80 8F 93

                       Tastatur-Codes für Funktionen (mit CTRL-ENTER angek.)

0269  FA 94 9E DF      DEFB   FAH,94H,9EH,DFH,BFH,E0H,F9H,83H      Bitreihe 0
      BF E0 F9 83
0271  F5 F4 A0 E1      DEFB   F5H,F4H,A0H,E1H,00H,D9H,D3H,00H      Bitreihe 1
      00 D9 D3 00
0279  00 00 00 00      DEFB   00H,00H,00H,00H,00H,01H,00H,00H      Bitreihe 2
      00 01 00 00
0281  F3 90 96 E3      DEFB   F3H,90H,96H,E3H,00H,DDH,D2H,C6H      Bitreihe 3
      00 DD D2 C6
0289  F7 F6 DB E2      DEFB   F7H,F6H,DBH,E2H,00H,D8H,CBH,00H      Bitreihe 4
      00 D8 CB 00
0291  F8 DE C1 E4      DEFB   F8H,DEH,C1H,E4H,00H,D7H,C9H,82H      Bitreihe 5
      00 D7 C9 82

                       Token, denen bei der Ausgabe das Zeichen '('
                       angefügt werden muß.

0299  E2 E1 E3 E4      DEFB   E2H,E1H,E3H,E4H,DFH,E0H,D7H
      DF E0 D7
02A0  DD D9 D8 F7      DEFB   DDH,D9H,D8H,F7H,F5H,F3H,F8H
      F5 F3 F8
02A7  F7 F9 9D F6      DEFB   F7H,F9H,9DH,F6H,F4H,DEH,E5H,FAH
      F4 DE E5 FA

                       ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                       Tabelle zur Ausgabe der Grafikzeichen auf einem Drucker.
                       Pro Zeichen enthält die Tabelle zwei Byte.

02AF  80 80            DEFB   80H,80H        ;Zeichen 80H
02B1  80 B8            DEFB   80H,B8H        ;Zeichen 81H
02B3  B8 80            DEFB   B8H 80H        ;Zeichen 82H
02B5  B8 B8            DEFB   B8H B8H        ;Zeichen 83H
02B7  80 87            DEFB   80H,87H        ;Zeichen 84H
02B9  80 BF            DEFB   80H,BFH        ;Zeichen 85H
02BB  B8 87            DEFB   B8H,87H        ;Zeichen 86H
02BD  B8 BF            DEFB   B8H,BFH        ;Zeichen 87H
02BF  87 80            DEFB   87H,80H        ;Zeichen 88H
02C1  87 B8            DEFB   87H,BFH        ;Zeichen 89H
02C3  BF 80            DEFB   BFH,80H        ;Zeichen 8AH
```

```
02C5  BF B8           DEFB   BFH,B8H          ;Zeichen 8BH
02C7  87 87           DEFB   87H,87H          ;Zeichen 8CH
02C9  87 BF           DEFB   87H,BFH          ;Zeichen 8DH
02CB  BF 87           DEFB   BFH,87H          ;Zeichen 8EH
02CD  BF BF           DEFB   BFH,BFH          ;Zeichen 8FH

                      ****************************************

                      Frequenz-Tabelle für das SOUND-Kommando
                      pro Note ein 2-Byte Eintrag.

02CF  72 02 4F 02     DEFW   626,591,558,526,497,469,439,414      A2 - E3
      2E 02 0E 02
      F1 01 D5 01
      B7 01 9E 01
02DF  86 01 70 01     DEFW   390,368,347,328,309,291,275,259      F3 - C4
      5B 01 48 01
      35 01 23 01
      13 01 03 01
02EF  F4 00 E6 00     DEFW   244,230,217,205,193,182,171,161      C#4 - G#4
      D9 00 CD 00
      C1 00 B6 00
      AB 00 A1 00
02FF  98 00 8F 00     DEFW   152,143,135,127,120,112,106          A4 - D#5
      87 00 7F 00
      78 00 70 00
      6A 00

                      ****************************************

                      Zeichen an Cursor-Position wiederherstellen
                      (Teil der Bildschirmausgabe-Routine)
0300  47              LD     B,A              ;auszugebendes Zeichen in B
0301  3A 3C 78        LD     A,(783CH)        ;Zeichen an Cursor-Position laden
0304  2A 20 78        LD     HL,(7820H)       ;Cursoradresse laden
0307  77              LD     (HL),A           ;Zeichen ausgeben
0308  78              LD     A,B              ;auszug. Zeichen wieder in A
0309  C9              RET

                      ****************************************

                      Cursor-Adresse eine Zeile zurück
                      Eing.: HL = aktuelle Cursoradresse
030A  01 20 00        LD     BC,32            ;Zeilenlänge
```

```
031A  B7              OR    A              ;Carry löschen
031B  ED 42           SBC   HL,BC          ;Cursoradr. - 1 Zeile
031D  22 20 78        LD    (7820H),HL     ;in Cursor-Pointer
0320  C9              RET

                      ****************************************

                      Multiplikatoren für Tondauer des SOUND-Kommandos
                      1 Byte pro möglichem Eingabe-Code (1-9)

0321  01 02 03 04     DEFB  1,2,3,4,6,8,12,16,24
      06 08 0C 10
      18

                      ****************************************

                      Zeichen auf Bildschirm, Drucker oder Kassette
                      ausgeben.
                      Eing.: A = auszugebendes Zeichen
                             789CH = Ausgabe-Flag (0=Bildschirm, 1=Drucker
                                                   80 = Kassette)
032A  C5              PUSH  BC             ;BC retten
032B  4F              LD    C,A            ;Zeichen in C merken
032C  CD C1 79        CALL  79C1H          ;RAM-Erweiterungsausg. (RET)
032F  3A 9C 78        LD    A,789CH        ;Ausgabe-Flag laden
0332  B7              OR    A              ;und testen
0333  79              LD    A,C            ;Zeichen zurück in A
0334  C1              POP   BC             ;BC wieder laden
0335  FA 54 3B        JP    M,3B54H        ;Kassette? ja - weiter bei 3B54H
0338  20 62           JR    NZ,039CH       ;Drucker? ja - zur Druckausgabe

                      ****************************************

                      Ein Zeichen auf dem Bildschirm ausgeben
                      Eing.: A = auszugebendes Zeichen
033A  D5              PUSH  DE             ;Register sichern
033B  F5              PUSH  AF
033C  C5              PUSH  BC
033D  E5              PUSH  HL
033E  CD 0B 30        CALL  300BH          ;Aufruf der Ausgaberoutine
0341  E1              POP   HL             ;Register wieder laden
0342  C1              POP   BC
0343  00              NOP
0344  00              NOP
```

```
0345  F1              POP     AF
0346  D1              POP     DE
0347  C9              RET

                      ****************************************

                      Cursorposition in Zeile ermitteln
                      bei LASER 110-310 nicht benutzt
0348  3A 3D 78        LD      A,(783DH)
034B  E6 08           AND     8
034D  3A 20 78        LD      A,(7820H)
0350  28 03           JR      Z,0355H
0352  0F              RRCA
0353  E6 1F           AND     1FH
0355  E6 1F           AND     1FH
0357  C9              RET

                      ****************************************

                      Tastatur-Abfrage
                      Ausg.: A = ASCII-Code oder 0
0358  CD C4 79        CALL    79C4H           ;RAM-Erweiter.ausgang (RET)
035B  D5              PUSH    DE              ;DE sichern
035C  CD 2B 00        CALL    002BH           ;Tastatur auswerten
035F  D1              POP     DE              ;DE wiederherstellen
0360  C9              RET

                      ****************************************

                      Tabelle der Zeitgrundwerte für jede einzelne
                      Note des SOUND-Kommandos.

0361  0A 0B 0C 0C     DEFB    10,11,12,12,13,14,15,15         ;A2 - E3
      0D 0E 0F 0F
0369  10 11 12 13     DEFB    16,17,18,19,21,22,23,25         ;F3 - C4
      15 16 17 19
0371  1A 1C 1D 1F     DEFB    26,28,29,31,33,35,37,39         ;C#4 - G#4
      21 23 25 27
0379  29 2C 2E 31     DEFB    41,44,46,49,52,53,58            ;A4 - D#5
      34 35 3A

                      ****************************************

                      OK- und Fehlermeldung des VERIFY-Kommandos
```

```
0380  4F 4B              DEFM   'OK'
0382  0D 00              DEFB   0DH,00H
0384  45 52 52 4F 52     DEFM   'ERROR'
0389  0D 00              DEFB   0DH,00H

                         ****************************************

                         Ausgabe-Flag auf Bildschirm.
                         CR auf Drucker, wenn nicht am Zeilenanfang
038B  AF                 XOR    A              ;Ausgabe-Flag auf Bildschirm
038C  32 9C 78           LD     (789CH),A
038F  3A 9B 78           LD     A,(789BH)      ;Druckerposition in Zeile
0392  B7                 OR     A              ;= 0?
0393  C8                 RET    Z              ;ja - fertig

                         Carriage-Return auf Drucker ausgeben
0394  3E 0D              LD     A,0DH          ;CR laden
0396  D5                 PUSH   DE             ;DE sichern
0397  CD 9C 03           CALL   039CH          ;CR ausgeben
039A  D1                 POP    DE             ;DE wieder herstellen
039B  C9                 RET

                         ****************************************

                         Zeichen auf Drucker ausgeben.
                         Eing.: A = auszugebendes Zeichen
                                789B = Druckkopfposition
039C  F5                 PUSH   AF             ;Register retten
039D  D5                 PUSH   DE
039E  C5                 PUSH   BC
039F  4F                 LD     C,A            ;Zeichen in C
03A0  1E 00              LD     E,0            ;E = 0
03A2  FE 0C              CP     0CH            ;Ist es ein Form-Feed?
03A4  28 10              JR     Z,03B6         ;ja!
03A6  FE 0A              CP     0AH            ;Ist es ein Line-Feed?
03A8  20 03              JR     NZ,03ADH       ;nein!
03AA  3E 0D              LD     A,0DH          ;ja, durch Carriage-Return ersetz.
03AC  4F                 LD     C,A            ;und in C
03AD  FE 0D              CP     0DH            ;Ist es ein Carriage-Return?
03AF  28 05              JR     Z,03B6H        ;ja!
03B1  3A 9B 78           LD     A,(789BH)      ;Druckkopfposition laden
03B4  3C                 INC    A              ;+1
03B5  5F                 LD     E,A            ;in E
```

```
03B6  7B           LD    A,E          ;neue Position speichern (CR=0)
03B7  32 9B 78     LD    (789BH),A
03BA  79           LD    A,C          ;auszug. Zeichen in A
03BB  CD 3B 00     CALL  003BH        ;Zeichen drucken
03BE  C1           POP   BC           ;Registerinhalte wiederherst.
03BF  D1           POP   DE
03C0  F1           POP   AF
03C1  C9           RET                ;fertig

                   ****************************************

                   Aufruf von Treiber-Routinen über den
                   Device-Control-Block
                   Eing.: DE = DCB-Adresse
                          B  = DCB-Typ, A = auszug. Zeichen (nur Ausgabe)
                          BC muß auf dem Stack sein
03C2  E5           PUSH  HL           ;Register sichern
03C3  DD E5        PUSH  IX
03C5  D5           PUSH  DE           ;DCB-Adresse in IX
03C6  DD E1        POP   IX
03C8  D5           PUSH  DE           ;und auf Stack
03C9  21 DD 03     LD    HL,03DDH     ;Rücksprungadresse auf Stack
03CC  E5           PUSH  HL
03CD  4F           LD    C,A          ;Zeichen nach C
03CE  1A           LD    A,(DE)       ;DCB-Kennung laden (1.Byte)
03CF  A0           AND   B            ;mit vorgegeb. Typ undieren
03D0  B8           CP    B            ;richtiger Typ?
03D1  C2 33 78     JP    NZ,7833H     ;nein, über RAM  7833H zurück
03D4  FE 02        CP    2            ;bei Eingaben Carry setzen
03D6  DD 6E 01     LD    L,(IX+1)     ;aus DCB Treiber-adresse laden
03D9  DD 66 02     LD    H,(IX+2)
03DC  E9           JP    (HL)         ;Treiber anspringen

                   Rückkehr vom Treiber
03DD  D1           POP   DE           ;Register wiederherstellen
03DE  DD E1        POP   IX
03E0  E1           POP   HL
03E1  C1           POP   BC
03E2  C9           RET                ;fertig

                   ****************************************

                   Einlesen einer Zeile von der Tastatur.
                   Zeile wird bis zur Betätigung der RETURN- oder BREAK-
```

```
                        Taste eingelesen, auf dem Bildschirm dargestellt und
                        anschließend in den I/O-Buffer übertragen.

                        Vorbereiten der Pointer und Flags
03E3  21 39 78          LD      HL,7839H        ;Initialisierungs-Flag für
03E6  CB EE             SET     5,(HL)          ;gepufferte Ausgabe setzen.
03E8  2A 20 78          LD      HL,(7820H)      ;Cursoradresse laden
03EB  CD 53 00          CALL    0053H           ;Zeichen an Cursorposition sichern
03EE  7C                LD      A,H             ;Cursor am Anfang der letzt.Zeile?
03EF  FE 71             CP      71H
03F1  20 10             JR      NZ,0403H        ;nein
03F3  7D                LD      A,L
03F4  FE E0             CP      0E0H
03F6  20 0B             JR      NZ,0403H        ;nein
03F8  3A D7 7A          LD      A,(7AD7H)       ;Status der 1.Zeile prüfen
03FB  B7                OR      A               ;= Folgezeile?
03FC  20 05             JR      NZ,0403H        ;nein!
03FE  3E 0D             LD      A,0DH           ;Bild eine Zeile hochrollen
0400  CD 8B 30          CALL    308BH
0403  41                LD      B,C             ;Länge Vorspanntext in B
0404  C5                PUSH    BC              ;auf Stack (B=C)
0405  21 39 78          LD      HL,7839H        ;Flag 2 adressieren
0408  CB 86             RES     0,(HL)          ;CR-Flag rücksetzen
040A  CB 96             RES     2,(HL)          ;BREAK-Flag rücksetzen
040C  CB 46             BIT     0,(HL)          ;warten, bis CR-Flag gesetzt
040E  28 FC             JR      Z,040CH

                        Anfangsadresse der Eingabezeile ermitteln
0410  3A A6 78          LD      A,(78A6H)       ;Spalte in Eingabezeile laden
0413  4F                LD      C,A             ;in BC
0414  AF                XOR     A
0415  32 A6 78          LD      (78A6H),A       ;Spaltenzähler = 0 (Zeilenanfang)
0418  47                LD      B,A
0419  2A 20 78          LD      HL,(7820H)      ;Cursoradresse laden
041C  ED 42             SBC     HL,BC           ;- Spalte = Zeileanfang
041E  22 20 78          LD      (7820H),HL      ;zurück in Cursor-Pointer

                        Buffer- und Zeilenadresse laden
0421  11 E8 79          LD      DE,79E8H        ;Anfangsadresse des I/O-Buffers
0424  C1                POP     BC              ;Zeichenzähler des Vorspanntextes
0425  21 39 78          LD      HL,7839H        ;Flag 2 adressieren
0428  CB 66             BIT     4,(HL)          ;Ist das ein INPUT-Kommando?
042A  2A 20 78          LD      HL,(7820H)      ;Anfangsadresse der Zeile laden
042D  28 42             JR      Z,0471H         ;kein INPUT-Cmd, weiter bei 471H
```

```
                        Bei INPUT Textpointer hinter vorgegebenen Text setzen
042F  C5          PUSH  BC              ;Register sichern
0430  E5          PUSH  HL
0431  CD A8 33     CALL  33A8H           ;Status der Zeile ermitteln
0434  E1          POP   HL              ;HL + BC wieder laden
0435  C1          POP   BC
0436  B7          OR    A               ;Folgezeile? (Status=00)
0437  20 08       JR    NZ,0441H        ;nein!
0439  7D          LD    A,L             ;Zeilenadresse in HL - 1 Zeile
043A  D6 20       SUB   32
043C  6F          LD    L,A
043D  7C          LD    A,H
043E  DE 00       SBC   A,0
0440  67          LD    H,A
0441  48          LD    C,B             ;Anzahl Vorspannzeichen
0442  1A          LD    A,(DE)          ;Pointer hinter Vorspanntext
0443  BE          CP    (HL)            ;vergleichen, ob nicht verändert
0444  20 07       JR    NZ,044DH        ;nicht gleich, aufhören
0446  23          INC   HL              ;Bildpointer + 1
0447  13          INC   DE              ;Bufferpointer + 1
0448  10 F8       DJNZ  0442H           ;fertig?
044A  C5          PUSH  BC              ;wenn gleich, Länge merken
044B  18 04       JR    0451H           ;weiter bei 0451
044D  01 00 00    LD    BC,0            ;ungleich, Länge = 0
0450  C5          PUSH  BC              ;auf den Stack
0451  E5          PUSH  HL              ;HL retten
0452  CD A8 33     CALL  33A8H           ;Status der Zeile lesen
0455  E1          POP   HL              ;HL + BC wieder laden
0456  C1          POP   BC
0457  C5          PUSH  BC              ;Länge wieder merken
0458  FE 80       CP    80H             ;Einzelzeile?
045A  28 0A       JR    Z,0466H         ;ja!
045C  3E 40       LD    A,64            ;Max. Zeichenzahl = 64 - Vorspann
045E  91          SUB   C
045F  47          LD    B,A
0460  D1          POP   DE              ;Anzahl Vorspannzeichen im Stack
0461  1E 00       LD    E,0             ;= 0 setzen
0463  D5          PUSH  DE
0464  18 05       JR    046BH           ;2 Zeilen übernehmen
0466  06 20       LD    B,32            ;1 Zeile übernehmen
0468  2A 20 78    LD    HL,(7820H)      ;Textanfangsadr. laden
046B  11 E8 79    LD    DE,79E8H        ;I/O-Bufferadresse
046E  C3 A8 3E    JP    3EA8H           ;Hintergrundfarbe prüfen
```

```
                                        ;bei grünem Hintergr.weiter b.04B8
                                        ;bei schw. Hintergr. weiter b.3E6A

                    Textanfangsadresse und max.Länge ermitteln,
                    wenn nicht INPUT-Kommando
0471  01 00 00      LD      BC,0          ;Vortextlänge = 0 setzen
0474  C5            PUSH    BC            ;auf Stack
0475  E5            PUSH    HL            ;HL retten
0476  CD A8 33      CALL    33A8H         ;Status der Zeile ermitteln
0479  E1            POP     HL            ;HL wieder laden
047A  FE 80         CP      80H           ;Einzelzeile?
047C  28 0E         JR      Z,048CH       ;ja!
047E  FE 81         CP      81H           ;2 Zeilen?
0480  28 06         JR      Z,0488H       ;ja!
0482  01 20 00      LD      BC,32         ;bei Folgezeile eine Zeile zurück
0485  B7            OR      A
0486  ED 42         SBC     HL,BC
0488  06 40         LD      B,64          ;2 Zeilen übernehmen
048A  18 02         JR      048EH
048C  06 20         LD      B,32          ;1 Zeile übernehmen
048E  3A 18 78      LD      A,(7818H)     ;Hintergrundfarbe prüfen
0491  B7            OR      A             ;0 = grün, 1 = schwarz
0492  CA 40 3E      JP      Z,3E40H       ;bei grün weiter bei 3E40H

                    Übertragen der Daten vom Bild zum I/O-Buffer
0495  7E            LD      A,(HL)        ;Zeichen vom Bild laden
0496  FE 40         CP      64            ;Grafik- oder Inverse?
0498  DA AE 04      JP      C,04AEH       ;nein, übernehmen
049B  C1            POP     BC            ;wenn nicht INPUT, dann sind
                                          ;Grafik und Inverse nur in
                                          ;Strings zugelassen
049C  11 A4 04      LD      DE,04A4H      ;Rücksprungadresse in Stack
049F  D5            PUSH    DE
04A0  C5            PUSH    BC
04A1  C3 02 05      JP      0502H         ;Textendekennung prüfen (BREAK?)

04A4  D8            RET     C             ;BREAK, zurück zum BASIC
04A5  21 1A 3E      LD      HL,3E1AH      ;Text "SYNTAX ERROR"
04A8  CD A7 28      CALL    28A7H         ;ausgeben
04AB  C3 E3 03      JP      03E3H         ;zurück zur Zeileneingabe

04AE  FE 22         CP      22H           ;Stringkennzeichnung?
04B0  20 31         JR      NZ,04E3       ;nein, weiter
04B2  12            LD      (DE),A        ;Zeichen in I/O-Buffer
```

```
04B3  23          INC   HL          ;Bildadresse + 1
04B4  13          INC   DE          ;Bufferadresse + 1
04B5  05          DEC   B           ;Zeichenzähler -1
04B6  28 36       JR    Z,04EEH     ;wenn 0, übernahme beenden

04B8  7E          LD    A,(HL)      ;Zeichen aus Bild laden
04B9  FE 40       CP    64          ;norm. Textzeichen?
04BB  DA C9 04    JP    C,04C9H     ;ja!
04BE  FE 80       CP    128         ;Inverses Textzeichen?
04C0  DA C5 04    JP    C,04C5H     ;ja!
04C3  E6 8F       AND   8FH         ;Grafikzeichen, Bits 4,5,6 löschen
04C5  F6 80       OR    80H         ;Bit 7 setzen
04C7  18 13       JR    04DCH

04C9  FE 22       CP    22H         ;Stringbegrenzer '"'?
04CB  20 09       JR    NZ,04D6H    ;nein!
04CD  E5          PUSH  HL          ;HL retten
04CE  21 39 78    LD    HL,7839H    ;Flag 2 adressieren
04D1  CB 66       BIT   4,(HL)      ;INPUT-Kommando?
04D3  E1          POP   HL          ;HL wieder laden
04D4  28 0D       JR    Z,04E3H     ;nein - ab jetzt Grafik u.
                                    ;Inverse nicht erlaubt.
04D6  CB 6F       BIT   5,A         ;Zeichen in echten ASCII-Code
04D8  20 02       JR    NZ,04DCH    ;umwandeln, z.B. 'A' von 01 in 41
04DA  F6 40       OR    40H         ;betrifft Codes 00 - 1FH
04DC  12          LD    (DE),A      ;Zeichen in I/O-Buffer
04DD  23          INC   HL          ;Bildadresse + 1
04DE  13          INC   DE          ;Bufferadresse + 1
04DF  10 D7       DJNZ  04B8H       ;Zähler - 1
04E1  18 0B       JR    04EEH       ;= 0, dann fertig

04E3  CB 6F       BIT   5,A         ;Zeichen in echten ASCII-Code
04E5  20 02       JR    NZ,04E9H    ;umwandeln, z.B. 'C' von 03 in 43
04E7  F6 40       OR    40H         ;betrifft Codes 00 -1FH
04E9  12          LD    (DE),A      ;Zeichen in I/O-Buffer
04EA  23          INC   HL          ;Bildadresse + 1
04EB  13          INC   DE          ;Bufferadresse + 1
04EC  10 A7       DJNZ  0495H       ;Zähler - 1

                  übertragung beendet, Bufferinhalt vervollständigen
04EE  1B          DEC   DE          ;Blanks am Bufferende elimin.
04EF  7A          LD    A,D         ;am Bufferanfang?
04F0  FE 79       CP    79H
04F2  20 06       JR    NZ,04FAH    ;nein
```

```
04F4  7B           LD    A,E
04F5  FE E8        CP    0E8H
04F7  DA FF 04     JP    C,04FFH       ;ja, fertig
04FA  1A           LD    A,(DE)        ;Zeichen laden
04FB  FE 20        CP    20H           ;= Blank?
04FD  28 EF        JR    Z,04EEH       ;ja, weiter zurück

04FF  13           INC   DE            ;Bufferende mit X'00'
0500  AF           XOR   A             ;kennzeichnen
0501  12           LD    (DE),A

                   Abhängig vom Zeilenstatus 1 oder 2 Leerzeilen ausg.
0502  CD A8 33     CALL  33A8H         ;Zeilenstatus ermitteln
0505  2A 20 78     LD    HL,(7820H)    ;Cursor-Pointer laden
0508  FE 81        CP    81H           ;2 Zeilen?
050A  CD 53 00     CALL  0053H         ;Zeichen aus Cursorposition sich.
050D  20 04        JR    NZ,0513H      ;einzeilig
050F  AF           XOR   A             ;1 Leerzeile ausgeben
0510  CD 8B 30     CALL  308BH
0513  AF           XOR   A             ;1 Leerzeile ausgeben
0514  CD 8B 30     CALL  308BH

0517  3A 38 78     LD    A,(7838H)     ;Flag 1 laden
051A  E6 FD        AND   0FDH          ;INVERSE-Flag rücksetzen
051C  32 38 78     LD    (7838H),A     ;Flag 1 wieder zurück
051F  21 39 78     LD    HL,7839H      ;Flag 2 adressieren
0522  CB 56        BIT   2,(HL)        ;BREAK-Flag gesetzt?
0524  28 05        JR    Z,052BH       ;nein!
0526  3E 01        LD    A,1           ;BREAK, A=1
0528  37          SCF                 ;+ Carry setzen
0529  18 01        JR    052CH
052B  AF           XOR   A             ;kein BREAK, A=0

052C  21 39 78     LD    HL,7839H      ;Flag 2 adressieren
052F  CB A6        RES   4,(HL)        ;INPUT-Cmd Flag zurücksetzen
0531  21 E8 79     LD    HL,79E8H      ;I/O-Buffer adressieren
0534  C1           POP   BC            ;auf Beginn der Eingabe
0535  F5          PUSH  AF            ;BREAK-Kenner sichern
0536  09          ADD   HL,BC
0537  C3 29 3E     JP    3E29H         ;weiter bei 3E29H
```

****

Teil der INPUT-Command Routine

```
                        Einlesen einer Zeile in den I/O-Buffer
053A  3A AF 7A          LD      A,(7AAFH)       ;Warten, bis Textausgabe abgeschl.
053D  B7                OR      A               ;7AAFH enthält Anzahl Zeichen im
053E  20 FA             JR      NZ,053AH        ;im Print-Buffer; bei 0 = leer

0540  06 40             LD      B,64            ;I/O-Buffer löschen (Länge 64)
0542  21 E8 79          LD      HL,79E8H        ;Bufferanfangs-Adresse
0545  3E 20             LD      A,' '           ;Leerzeichen in A
0547  77                LD      (HL),A          ;in Buffer übertragen
0548  23                INC     HL              ;Bufferadresse + 1
0549  10 FC             DJNZ    0547H           ;Zähler - 1, wenn 0 - fertig!
054B  AF                XOR     A               ;0 in A
054C  77                LD      (HL),A          ;Bufferende mit X'00' kennzeichn.

054D  CD A8 33          CALL    33A8H           ;Zeilenstatus ermitteln
0550  B7                OR      A               ;Folgezeile?
0551  3A A6 78          LD      A,(78A6H)       ;Spaltenzähler laden
0554  20 02             JR      NZ,0558H        ;keine Folgezeile!
0556  C6 20             ADD     A,32            ;bei Folgezeile eine Zeile addier.
0558  4F                LD      C,A             ;Spaltenzähler in BC übergeben
0559  AF                XOR     A               ;B dazu = 0 setzen
055A  47                LD      B,A
055B  2A 20 78          LD      HL,(7820H)      ;Cursor-Pointer laden
055E  ED 42             SBC     HL,BC           ;- Spalte = Zeilenanfang
0560  11 E8 79          LD      DE,79E8H        ;I/O-Buffer-Adresse laden
0563  C5                PUSH    BC              ;Spaltenzähler merken
0564  ED B0             LDIR                    ;vorh. Text aus Zeile in Buffer
0566  C1                POP     BC              ;Spaltenzähler wieder laden
0567  21 39 78          LD      HL,7839H        ;Flag 2 adressieren
056A  CB E6             SET     4,(HL)          ;INPUT-Cmd Flag setzen
056C  CD E3 03          CALL    03E3H           ;Zeile einlesen
056F  C9                RET

                        ****************************************

                        RUN-Kommando für autom. Start bei CRUN
0570  52 55 4E          DEFM    'RUN'
0573  00                DEFB    0

                        ****************************************

                        Drucker - Treiber

058D  79                LD      A,C             ;auszugebendes Zeichen laden
```

```
058E  B7          OR    A             ;= leer?
058F  28 33       JR    Z,05C4H       ;ja, nur Druckerstatus ermitteln
0591  FE 0B       CP    0BH           ;Seitenvorschub?
0593  28 0A       JR    059FH         ;ja - ausführen
0595  FE 0C       CP    0CH           ;bedingter Seitenvorschub?
0597  20 14       JR    NZ,05ADH      ;nein!
0599  AF          XOR   A             ;wird nur ausgeführt, wenn Anzahl
059A  DD B6 03    OR    (IX+3)        ;Zeilen/Seite ungleich 0 ist
059D  28 0E       JR    Z,05ADH       ;Sonst Ausgabe 0C auf Drucker
059F  DD 7E 03    LD    A,(IX+3)      ;Zeilen/Seite
05A2  DD 96 04    SUB   (IX+4)        ;- Anzahl gedruckter Zeilen
05A5  47          LD    B,A           ;in B als Vorschubzähler
05A6  CD E2 3A    CALL  3AE2H         ;Carriage-Return + Line Feed ausg.
05A9  10 FB       DJNZ  05A6H         ;bis zur neuen Seite
05AB  18 12       JR    05BFH

05AD  CD B6 3A    CALL  3AB6H         ;Zeichen ausgeben
05B0  79          LD    A,C           ;Zeichen nochmals laden
05B1  FE 0D       CP    0DH           ;war das ein CR?
05B3  C0          RET   NZ            ;nein, fertig
05B4  DD 34 04    INC   (IX+4)        ;Zeilenzähler im DCB + 1
05B7  DD 7E 04    LD    A,(IX+4)      ;am Anfang einer neuen Seite?
05BA  DD BE 03    CP    (IX+3)        ;(Zeilenzähler - Zeilen/Seite)
05BD  79          LD    A,C           ;Zeichen wieder in A
05BE  C0          RET   NZ            ;keine neue Seite - fertig
05BF  DD 36 04 00 LD    (IX+4),0      ;Zeilenzähler = 0
05C3  C9          RET

05C4  DB 00       IN    A,(0)         ;Drucker-Status ermitteln
05C6  E6 01       AND   1             ;nur BUSY wird geprüft
05C8  C9          RET

                  ************************************

                  4-Byte Druckpuffer für Grafikdruck löschen
05C9  C5          PUSH  BC            ;BC + HL retten
05CA  E5          PUSH  HL
05CB  06 04       LD    B,4           ;Zähler = 4
05CD  21 D2 7A    LD    HL,7AD2H      ;Pufferadresse laden
05D0  77          LD    (HL),A        ;A in Puffer übertragen
05D1  23          INC   HL            ;Pufferadresse + 1
05D2  10 FC       DJNZ  05D0H         ;Zähler -1 = 0? ja - fertig!
05D4  E1          POP   HL            ;Register wieder herstellen
05D5  C1          POP   BC
```

```
05D6  C9                RET

                        ************************************

                        Teil der Tastaturabfrage
                        Behandelt das Betätigen einer zweiten Taste, bevor
                        die erste losgelassen wurde (Rollover)
                        Im Flag 1 (7838H) werden die Bits 3 und 4 benutzt, um
                        den Status der zwei Tastaturpuffer B1 (7836H) und
                        B2 (7837H) anzuzeigen.

                        Bit4  Bit3    Status
                         0     0      B1 und B2 werden nicht gedrückt
                         0     1      B1 gedrückt, B2 nicht gedrückt
                         1     0      B1 nicht gedrückt, B2 gedrückt
                         1     1      B1 und B2 gedrückt

05D7  21 38 78          LD    HL,7838H      ;Flag 1 adressieren
05DA  CB 56             BIT   2,(HL)        ;Funktions-Flag gesetzt?
05DC  28 15             JR    Z,05F3H       ;nein - weiter bei 05F3H
05DE  57                LD    D,A           ;Tastencode sichern
05DF  3A 3A 78          LD    A,(783AH)     ;Zeitwert laden
05E2  B7                OR    A             ;= 0 ?
05E3  28 0F             JR    Z,05F4H       ;ja - weiter bei 05F4H
05E5  3C                INC   A             ;Zeitwert + 1
05E6  32 3A 78          LD    (783AH),A     ;zurückspeichern
05E9  FE 2A             CP    42            ;Zeit abgelaufen? (ca. 0.84 sec)
05EB  28 02             JR    Z,05EFH       ;ja!
05ED  AF                XOR   A             ;Zeichen löschen
05EE  C9                RET                 ;und zurück
05EF  CB 96             RES   2,(HL)        ;Funktions-Flag löschen
05F1  AF                XOR   A             ;Zeichen löschen
05F2  C9                RET                 ;zurück

05F3  57                LD    D,A           ;Zeichen in D sichern
05F4  21 38 78          LD    HL,7838H      ;Flag 1 adressieren
05F7  7E                LD    A,(HL)        ;in A laden
05F8  E6 18             AND   00011000B     ;Bits 3 und 4 testen
05FA  20 0B             JR    NZ,0607H      ;Bit 3 und/oder Bit 4 gesetzt
05FC  CB DE             SET   3,(HL)        ;Bit 3 setzen
05FE  AF                XOR   A             ;B2 löschen
05FF  32 37 78          LD    (7837H),A
0602  7A                LD    A,D           ;Zeichen wieder laden
0603  32 36 78          LD    (7836H),A     ;und in B1 eintragen
```

```
0606  C9              RET             ;nur eine Taste gedrückt - fertig!

                      Taste wurde gehalten
0607  CB 66           BIT   4,(HL)      ;bereits zwei Tasten im Puffer?
0609  C2 2A           JP    NZ,0635H    ;ja!
060B  3A 36 78        LD    A,(7836H)   ;Zeichen aus B1 laden
060E  BA              CP    D           ;= gedrückte Taste?
060F  20 21           JR    NZ,0632H    ;nein, eine neue
0611  ED 4B 42 78     LD    BC,(7842H)  ;Zeilen-/Spaltenzähler laden
0615  2A 44 78        LD    HL,(7844H)  ;Matrixadresse laden
0618  7B              LD    A,E         ;Inhalt der Matrixzeile
0619  CD 35 2F        CALL  2F35H       ;restliche Tasten prüfen
061C  BA              CP    D           ;gleiche wie vorher?
061D  CA D7 2F        JP    Z,2FD7H     ;ja, zur Zeichenwiederholung
0620  FE 00           CP    0           ;keine weitere?
0622  CA D7 2F        JP    Z,2FD7H     ;ja, zur Zeichenwiederholung
0625  21 38 78        LD    HL,7838H    ;Flag 1 adressieren
0628  CB DE           SET   3,(HL)      ;Beide Statusbits 3+4 setzen
062A  CB E6           SET   4,(HL)
062C  CB 96           RES   2,(HL)      ;Funktions-Flag rücksetzen
062E  32 37 78        LD    (7837H),A   ;Zeichen in B2
0631  C9              RET               ;und zurück

0632  7A              LD    A,D         ;neuen Tastencode in A
0633  18 F0           JR    0625H       ;in B2 eintragen

                      Zwei Tasten bereits registriert
0635  3A 36 78        LD    A,(7836H)   ;Zeichen aus B1 laden
0638  BA              CP    D           ;= neuer Tastencode?
0639  28 08           JR    Z,0643H     ;ja!
063B  3A 37 78        LD    A,(7837H)   ;Zeichen aus B2 laden
063E  BA              CP    D           ;= neuer Tastencode?
063F  28 02           JR    Z,0643H     ;ja!
0641  AF              XOR   A           ;3 Tasten - igitt
0642  C9              RET               ;zurück mit A = 0

0643  ED 4B 42 78     LD    BC,(7842H)  ;Zeilen-/Spaltenzähler laden
0647  2A 44 78        LD    HL,(7844H)  ;Matrixadresse laden
064A  7B              LD    A,E         ;Inhalt der Matrixzeile laden
064B  CD 35 2F        CALL  2F35H       ;Matrix weiter durchsuchen
064E  BA              CP    D           ;gleiche Taste?
064F  28 05           JR    Z,0656H     ;ja!
0651  FE 00           CP    0           ;keine weitere Taste?
0653  C2 D7 2F        JP    NZ,2FD7H    ;ja - zur Zeichenwiederholung
```

```
0656  21 38 78      LD    HL,7838H      ;Flag 1 adressieren
0659  CB DE         SET   3,(HL)        ;Statusflag für B1 setzen
065B  CB A6         RES   4,(HL)        ;Statusflag für B2 löschen
065D  3A 36 78      LD    A,(7836H)     ;Zeichen aus B1 laden
0660  BA            CP    D             ;= eingegebenes Zeichen?
0661  20 05         JR    NZ,0668H      ;nein!
0663  AF            XOR   A             ;B2 löschen
0664  32 37 78      LD    (7837H),A
0667  C9            RET                 ;zurück

0668  3A 37 78      LD    A,(7837H)     ;B2 nach B1 übertragen
066B  32 36 78      LD    (7836H),A
066E  18 F3         JR    0663H         ;B2 löschen
```

****

BASIC - Initialisierung Teil 1

```
0674  00            NOP                 ;fängt gut an
0675  00            NOP
0676  21 D2 06      LD    HL,06D2H      ;ROM 6D2 - 707 in
0679  11 00 78      LD    DE,7800H      ;RAM 7800 - 7835
067C  01 36 00      LD    BC,36H        ;übertragen
067F  ED B0         LDIR
0681  3D            DEC   A             ;das ganze 128x
0682  3D            DEC   A             ;warum ???????
0683  20 F1         JR    NZ,0676H      ;wahrscheinlich einbrennen !!!!!
0685  06 27         LD    B,39          ;die nächsten 39 Bytes löschen
0687  12            LD    (DE),A        ;(7836 - 785C)
0688  13            INC   DE
0689  10 FC         DJNZ  0687H
068B  C3 75 00      JR    0075H         ;zur BASIC - Initialisierung T. 2
```

****

BASIC - Initialisierung Teil 3

```
                    prüfen, ob ROM-Kassette vorhanden
068E  21 00 40      LD    HL,4000H      ;1. Möglichkeit bei 4000H
0691  CD A4 06      CALL  06A4H         ;dort prüfen
0694  21 00 60      LD    HL,6000H      ;2. Möglichkeit bei 6000H
0697  CD A4 06      CALL  06A4H         ;prüfen
069A  21 00 80      LD    HL,8000H      ;3. Möglichkeit bei 8000H
069D  CD A4 06      CALL  06A4H         ;prüfen
```

```
06A0  FB           EI              ;kein Einschub - Interrupts ein
06A1  C3 19 1A     JP    1A19H     ;zur BASIC - Hauptschleife

06A4  3E AA        LD    A,0AAH    ;ROM-Einschub muß mit der
06A6  BE           CP    (HL)      ;Bytefolge  AA 55 E7 18  beginnen
06A7  23           INC   HL        ;nächstes Byte
06A8  C0           RET   NZ        ;war schon nichts
06A9  2F           CPL             ;2. Wert (55) bilden
06AA  BE           CP    (HL)      ;gleich?
06AB  23           INC   HL        ;nächstes Byte
06AC  C0           RET   NZ        ;ungleich!
06AD  3E E7        LD    A,0E7H    ;3. Wert = E7
06AF  BE           CP    (HL)      ;stimmt der?
06B0  23           INC   HL        ;nächstes Byte
06B1  C0           RET   NZ        ;nein, auch nicht
06B2  2F           CPL             ;4. Wert (18) bilden
06B3  BE           CP    (HL)      ;stimmt dieser auch?
06B4  23           INC   HL        ;nächstes Byte
06B5  C0           RET   NZ        ;nein - kein Einschub
06B6  FB           EI              ;Interrupts einschalten
06B7  E9           JP    (HL)      ;ROM-Einschub anspringen

06CC  01 18 1A     LD    BC,1A18H  ;Adresse der Hauptschleife laden
06CF  C3 AE 19     JP    19AEH     ;BASIC-Variablen und Pointer init.

                   ****************************************

                   Der folgende Bereich von 6D2 bis 707 wird
                   in den RAM-Bereich von 7000 bis 7035 übertragen

                   Restart-Vektoren
06D2  C3 96 1C     JP    1C96H     ;RST 8H (Vergleich 1 Zeichen)
06D5  C3 78 1D     JP    1D78H     ;RST 10H (Nächster Zeichen)
06D8  C3 90 1C     JP    1C90H     ;RST 18H (Vergleich HL/DE)
06DB  C3 D9 25     JP    25D9H     ;RST 20H (Datentyp testen)
06DE  C9 00 00     RET             ;RST 28H
06E1  C9 00 00     RET             ;RST 30H
06E4  FB           EI              ;RST 38H (Interrupt)
06E5  C9 00        RET

                   Tastatur - Device-Control-Block
06E7  01           DEFB  1         ;DCB-Typ
06E8  F4 2E        DEFW  2EF4H     ;Adresse des Treibers
06EA  00 00 00
```

```
06ED  4B 49           DEFM   'KI'

                      Bildschirm - Device-Control-Block
                      bis auf die Cursor-Adresse nicht benutzt.
06EF  00              DEFB   0              ;DCB-Typ (unbekannt)
06F0  00 00           DEFW   0              ;von SET, RESET u. POINT benutzt.
06F2  00 70           DEFW   7000H          ;Cursor-Adresspointer
06F4  00 00 00

                      Drucker - Device-Control-Block
06F7  06              DEFB   6              ;DCB-Typ
06F8  8D 05           DEFW   058DH          ;Treiber-Adresse
06FA  43              DEFB   67             ;Zeilen/Seite +1
06FB  00              DEFB   0              ;Zeilenzähler
06FC  00
06FD  50 52           DEFM   'PR'

06FF  C3 00 50        JP     5000H          ;nicht benutzt
0702  C7 00 00        RST    0              ;nicht benutzt

0705  3E 00           LD     A,0            ;Ansprung bei falschem DCB-Typ
0707  C9              RET                   ;in der DCB-Aufrufroutine


                      ****************************************

                      Addition und Subtraktion mit einfacher Genauigkeit

                      Verschiedene Ansprungpunkte entspr. der geforderten
                      Funktion.
                      Eing.: X = Summand bzw. Subtrahend
                             HL bzw. Y  Summand bzw. Minuend
                      Ausg.: X = Summe bzw. Differenz

                      X = X + 0.5
0708  21 80 13        LD     HL,1380H       ;Adresse der Konstante 0.5
                      ----------------------
                      X = Konstante + X
070B  CD C2 09        CALL   09C2H          ;Konstante nach Y laden
070E  18 06           JR     0716H          ;Sprung zur Addition
                      ----------------------
                      X = Konstante - X
0710  CD C2 09        CALL   09C2H          ;Konstante nach Y
                      ----------------------
```

```
                        X = Y - X
0713  CD 82 09          CALL    0982H           ;X = -X
                        ------------------------

                        X = Y + X
0716  78                LD      A,B             ;Y = 0? (Exp. Y = 0)
0717  B7                OR      A
0718  C8                RET     Z               ;Ja, fertig
0719  3A 24 79          LD      A,(7924H)       ;X = 0? (Exp. X = 0)
071C  B7                OR      A
071D  CA B4 09          JP      Z,09B4H         ;Ja, fertig, X=Y
0720  90                SUB     B               ;Exp. X - Exp. Y in A
                                                ;Exp. Y <= Exp. X?
0721  30 0C             JR      NC,072FH        ;Ja
0723  2F                CPL                     ;Exp.Diff negieren
0724  3C                INC     A               ;X mit Y vertauschen
0725  EB                EX      DE,HL           ;LSB Y sichern
0726  CD A4 09          CALL    09A4H           ;X auf Stack bringen
0729  EB                EX      DE,HL           ;LSB Y wiederherstellen
072A  CD B4 09          CALL    09B4            ;Y nach X übertragen
072D  C1                POP     BC              ;Stack nach Y laden
072E  D1                POP     DE
072F  FE 19             CP      25              ;Exp.Diff > Mantisse (24 Bits)
0731  D0                RET     NC              ;Nein, X = X, fertig
0732  F5                PUSH    AF              ;Exp.Diff. sichern
0733  CD DF 09          CALL    09DFH           ;Vorzeichen-Bits = 1 setzen.
                                                ;A(7) = 1 wenn gleiche Vorzeichen
                                                ;A(7) = 0 bei ungleichen Vorzeichen
0736  67                LD      H,A             ;Vorzeichen-Flag sichern
0737  F1                POP     AF              ;Exp.Differenz zurückladen
0738  CD D7 07          CALL    07D7H           ;Y um diese Differenz rechts schieben
073B  B4                OR      H               ;Vorzeichen gleich?
073C  21 21 79          LD      HL,7921H        ;LSB X-Adresse in HL
073F  F2 54 07          JP      P,0754H         ;Nein, subtrahieren
                        ------------------------
                        Addition der Mantissen
0742  CD B7 07          CALL    07B7H           ;Mantissen addieren
0745  D2 96 07          JP      NC,0796H        ;Überlauf? Nein=Sprung
0748  23                INC     HL              ;Zeiger auf Exp. X
0749  34                INC     (HL)            ;Exp. X + 1
074A  CA B2 07          JP      Z,07B2H         ;Überlauf? Ja=OV-Error
074D  2E 01             LD      L,1             ;Mantisse von 1 um 1 Bit
074F  CD EB 07          CALL    07EBH           ;rechts schieben
0752  18 42             JR      0796H           ;fertig!
                        ------------------------
```

```
                        Subtraktion der Mantissen
0754  AF                XOR   A             ;Mant. Y - Mant. X nach Mant. Y
0755  90                SUB   B             ;niederw. Byte (durch Schieben entst.)
0756  47                LD    B,A           ;Ergebnis
0757  7E                LD    A,(HL)        ;LSB-Subtraktion
1758  9B                SBC   A,E
0759  5F                LD    E,A
075A  23                INC   HL            ;nächstes Byte
075B  7E                LD    A,(HL)        ;subtrahieren
075C  9A                SBC   A,D
075D  57                LD    D,A
075E  23                INC   HL            ;MSB subtrahieren
075F  7E                LD    A,(HL)
0760  99                SBC   A,C
0761  4F                LD    C,A           ;Unterlauf?
0762  DC C3 07          CALL  C,07C3H       ;Ja! Mant. Y negieren
                                            ;Vorzeichen-Flag invertieren

                        ########################################

                        Normalisieren
0765  68                LD    L,B           ;Erg. Mant. von CDEB nach CDHL
0766  63                LD    H,E
0767  AF                XOR   A             ;Schiebezähler = 0
0768  47                LD    B,A
0769  79                LD    A,C           ;MSB Y = 0?
076A  B7                OR    A
076B  20 18             JR    NZ,0785H      ;nein
076D  4A                LD    C,D           ;Y um 1 Byte Links schieben
076E  54                LD    D,H           ;H nach D
076F  65                LD    H,L           ;L nach H
0770  6F                LD    L,A           ;L = 0
0771  78                LD    A,B           ;Schiebezähler - 8
0772  D6 08             SUB   8
0774  FE E0             CP    0E0H          ;32 Linksschiebungen? (Zahl = 0)
0776  20 F0             JR    NZ,0768H      ;nein!
                        ----------------------
                        Reelle Zahl = 0 setzen
0778  AF                XOR   A             ;Exponent in Y = 0
0779  32 24 79          LD    (7924H),A     ;d.h. Y = 0
077C  C9                RET
                        ----------------------
                        2. Teil der Normalisierung
077D  05                DEC   B             ;Schiebezähler - 1
```

```
077E  29           ADD   HL,HL        ;CDHL ein Bit links (HL * 2)
077F  7A           LD    A,D         ;D * 2
0780  17           RLA
0781  57           LD    D,A
0782  79           LD    A,C         ;C * 2
0783  8F           ADC   A,A
0784  4F           LD    C,A         ;höchstes Bit Y gesetzt?
0785  F2 7D 07     JP    P,077DH     ;nein, weiter
0788  78           LD    A,B         ;Schiebezähler nach A
0789  5C           LD    E,H         ;CDHL wieder nach CDEB
078A  45           LD    B,L
078B  B7           OR    A           ;keine Verschiebung?
078C  28 08        JR    Z,0796H     ;ja
078E  21 24 79     LD    HL,7924H    ;Adresse X-Exponent
0791  86           ADD   A,(HL)      ;Exp. X + Anzahl Verschiebungen
0792  77           LD    (HL),A      ;= Exp. X. Unterlauf?
0793  30 E3        JR    NC,0778H    ;ja! X=0 und zurück
0795  C8           RET   Z           ;Anzahl Verschieb. = Exp. X?  zurück!
0796  78           LD    A,B         ;LSB Y laden
0797  21 24 79     LD    HL,7924H    ;Adresse X-Exponent
079A  B7           OR    A           ;LSB Y(7) = 0?
079B  FC A8 07     CALL  M,07A8H     ;Nein - Y runden
079E  46           LD    B,(HL)      ;Exp. X nach Exp. Y
079F  23           INC   HL          ;Vorzeichen Flag
07A0  7E           LD    A,(HL)      ;laden
07A1  E6 80        AND   80H         ;Vorzeichen ausblenden
07A3  A9           XOR   C           ;Mit MSB Y verknüpfen (invertieren)
07A4  4F           LD    C,A         ;und zurück nach MSB Y
07A5  C3 B4 09     JP    09B4H       ;Y nach X als Ergebnis

                   ****************************************

                   Runden
07A8  1C           INC   E           ;LSB Y + 1
07A9  C0           RET   NZ          ;= 0?, nein-fertig
07AA  14           INC   D           ;n.Byte Y + 1
07AB  C0           RET   NZ          ;= 0?, nein-fertig
07AC  0C           INC   C           ;MSB Y + 1
07AD  C0           RET   NZ          ;= 0?, nein-fertig
07AE  0E 80        LD    C,80H       ;ja, MSB Y = 80H
07B0  34           INC   (HL)        ;Exponent X + 1
07B1  C0           RET   NZ          ;= 0?, nein-zurück
                   ------------------
                   OVERFLOW-Error
```

```
07B2  1E 0A          LD    E,10          ;Fehlernummer in E
07B4  C3 A2 19       JP    19A2H         ;zur Fehlerroutine

                     ************************************

                     Mantissen-Addition einfacher Genauigkeit
                     Eing.: Mantisse X = Summand
                            Mantisse Y = Summand
                            HL = Adresse LSB X
                     Ausg.: Mantisse Y = Summe

07B7  7E             LD    A,(HL)        ;LSB X in A
07B8  83             ADD   A,E           ;+ LSB Y
07B9  5F             LD    E,A           ;Summe in LSB Y
07BA  23             INC   HL            ;X-Adresse + 1
07BB  7E             LD    A,(HL)        ;nächstes Byte addieren
07BC  8A             ADC   A,D
07BD  57             LD    D,A
07BE  23             INC   HL            ;HL = MSB X
07BF  7E             LD    A,(HL)        ;MSB X + MSB Y
07C0  89             ADC   A,C
07C1  4F             LD    C,A           ;in MSB Y
07C2  C9             RET

                     ************************************

                     Mantisse Y negieren
07C3  21 25 79       LD    HL,7925H      ;Vorzeichen-Flag invertieren
07C6  7E             LD    A,(HL)
07C7  2F             CPL
07C8  77             LD    (HL),A
07C9  AF             XOR   A             ;A = 0
07CA  6F             LD    L,A           ;L = 0
07CB  90             SUB   B             ;LSB Y = 0 - LSB Y
07CC  47             LD    B,A
07CD  7D             LD    A,L           ;A = 0
07CE  9B             SBC   A,E           ;n.Byte Y = 0 - n.Byte Y
07CF  5F             LD    E,A
07D0  7D             LD    A,L           ;A = 0
07D1  9A             SBC   A,D           ;n.Byte Y = 0 - n.Byte Y
07D2  57             LD    D,A
07D3  7D             LD    A,L           ;A=0
07D4  99             SBC   A,C           ;MSB Y = 0 - MSB Y
07D5  4F             LD    C,A
```

```
07D6  C9              RET

                      ****************************************

                      Zahl einfacher Genauigkeit nach rechts schieben
                      Eing.: Y = Zahl
                             A = Anzahl Verschiebungen
                      Ausg.: Y = Ergebnis
                             B = zus. niederwertiges Byte
07D7  06 00           LD     B,0            ;LSB des Ergebnisses = 0
07D9  D6 08           SUB    8              ;8 oder mehr Stellen schieben?
07DB  38 07           JR     C,07E4H        ;nein!
07DD  43              LD     B,E            ;Y um ein Byte nach rechts
07DE  5A              LD     E,D
07DF  51              LD     D,C
07E0  0E 00           LD     C,0
07E2  18 F5           JR     07D9H

07E4  C6 09           ADD    A,9            ;Anzahl Verschiebungen + 1 in L
07E6  6F              LD     L,A
07E7  AF              XOR    A              ;Carry löschen
07E8  2D              DEC    L              ;Schiebzähler - 1
07E9  C8              RET    Z              ;= 0? ja-fertig
07EA  79              LD     A,C            ;MSB Y ein Bit nach rechts
07EB  1F              RRA
07EC  4F              LD     C,A
07ED  7A              LD     A,D            ;n.Byte Y ein Bit nach rechts
07EE  1F              RRA
07EF  57              LD     D,A
07F0  7B              LD     A,E            ;n.Byte Y ein Bit nach rechts
07F1  1F              RRA
07F2  5F              LD     E,A
07F3  78              LD     A,B            ;LSB Y ein Bit nach rechts
07F4  1F              RRA
07F5  47              LD     B,A
07F6  18 EF           JR     07E7H          ;weiter

                      ****************************************

                      Konstanten

07F8  00 [illegible] 00 81                   ;= 1

                      für LOG - Funktion
```

```
07FC  03                                  ;Anzahl Konstanten = 3
07FD  AA 56 19 80                         ;= 0.598979
0801  F1 22 76 80                         ;= 0.961471
0805  45 AA 38 82                         ;= 2.88539

                        ****************************************

                        LOG - Funktion
                        Berechnet den natürlichen Logarithmus
                        Eing.: X = Argument
                        Ausg.: X = Ergebnis
0809  CD 55 09          CALL    0955H     ;Argument <= 0?
080C  B7                OR      A
080D  EA 4A 1E          JP      PE,1E4AH  ;ja, Function-Code Error
0810  21 24 79          LD      HL,7924H  ;Exponent d. Arguments in A
0813  7E                LD      A,(HL)
0814  01 35 80          LD      BC,8035H  ;Y = 0.707092
0817  11 F3 04          LD      DE,04F3H
081A  90                SUB     B         ;Offset Exp X in A
081B  F5                PUSH    AF        ;sichern
081C  70                LD      (HL),B    ;Exp. X = 0
081D  D5                PUSH    DE        ;Y auf Stack
081E  C5                PUSH    BC

                        X = (Arg - SQR(2)/2) / (Arg + SQR(2)/2)
081F  CD 16 07          CALL    0716H     ;X = X + 0.707092
0822  C1                POP     BC        ;Y wieder mit Konstante laden
0823  D1                POP     DE
0824  04                INC     B         ;Exp. Y + 1  (Y = SQR(2))
0825  CD A2 08          CALL    08A2H     ;X = SQR(2) / X
0828  21 F8 07          LD      HL,07F8H  ;Adresse der Konstante 1 laden
082B  CD 10 07          CALL    0710H     ;X = 1 - X

                        Reihe berechnen
082E  21 FC 07          LD      HL,07FCH  ;Adresse der 1. Reihenkonstante
0831  CD 9A 14          CALL    149AH     ;Reihe berechnen

0834  01 80 80          LD      BC,8080H  ;Y = -0.5
0837  11 00 00          LD      DE,0
083A  CD 16 07          CALL    0716H     ;X = X - 0.5
083D  F1                POP     AF        ;Exp. d. Arguments
083E  CD 89 0F          CALL    0F89H     ;X = X + A

                        X = X * LOG (2)
```

```
0841  01 31 80        LD      BC,8031H        ;Y = LOG(2)  ca. 0.693147
0844  11 18 72        LD      DE,7218H

                      ****************************************

                      Multiplikation mit einfacher Genauigkeit

                      X = X * Y
0847  CD 55 09        CALL    0955H           ;X = 0?
084A  C8              RET     Z               ;ja, fertig
084B  2E 00           LD      L,0             ;Flag für Exp.verarbeitung
084D  CD 14 09        CALL    0914H           ;Exponenten u. Vorzeichen verarb.
0850  79              LD      A,C             ;Mantisse aus Y nach 79AFH ....
0851  32 4F 79        LD      (794FH),A       ;MSB
0854  EB              EX      DE,HL           ;LSB
0855  22 50 79        LD      (7950H),A
0858  01 00 00        LD      BC,0            ;Y = Erg.register  löschen
0859  50              LD      D,B
085C  58              LD      E,B
085D  21 65 07        LD      HL,0765H        ;Untersch. Rücksprungadressen für
                                              ;3 Durchläufe auf Stack
0860  E5              PUSH    HL              ;nach 3. Durchl. zur Normalisierung
0861  21 69 08        LD      HL,0869H        ;nach 1. u. 2. Durchlauf
0864  E5              PUSH    HL              ;wiederholen
0865  E5              PUSH    HL
0866  21 21 79        LD      HL,7921H        ;Adresse LSB X laden
0869  7E              LD      A,(HL)          ;LSB X in A
086A  23              INC     HL              ;nächstes X-Byte adressieren
086B  B7              OR      A               ;Inhalt = 0 ?
086C  28 24           JR      Z,0892H         ;ja, Ergebnis 1 Byte rechts schieb.
086E  E5              PUSH    HL              ;Adresspointer sichern
086F  2E 08           LD      L,8             ;Bitzähler = 8
0871  1F              RRA                     ;Ein Bit in Carry schieben
0872  67              LD      H,A             ;A in H sichern
0873  79              LD      A,C             ;MSB des Ergebnisses laden
0874  D2 0B           JR      NC,0881H        ;Bit in Carry = 1 ?
0876  E5              PUSH    HL              ;ja! - HL retten
0877  2A 50 79        LD      HL,(7950H)      ;LSB 2.Faktor laden
087A  19              ADD     HL,DE           ;+ Ergebnis-LSB
087B  EB              EX      DE,HL           ;in LSB Y
087C  E1              POP     HL              ;HL wieder laden
087D  3A 4F 79        LD      A,(794FH)       ;MSB 2.Faktor laden
0880  89              ADC     A,C             ;+ Ergebnis-MSB
0881  1F              RRA                     ;Ergebnis 1 Bit rechts schieben
```

```
0882  4F          LD    C,A          ;MSB
0883  7A          LD    A,D          ;n. Byte
0884  1F          RRA
0885  57          LD    D,A
0886  7B          LD    A,E          ;n. Byte
0887  1F          RRA
0888  5F          LD    E,A
0889  78          LD    A,B          ;LSB
088A  1F          RRA
088B  47          LD    B,A
088C  2D          DEC   L            ;Bitzähler - 1
088D  7C          LD    A,H          ;X-Byte wieder laden
088E  20 E1       JR    NZ,0871H     ;Bitzähler = 0, nein-zurück
0890  E1          POP   HL           ;ja - X-Byteadresse laden
0891  RET                           ;weiter

0892  43          LD    B,E          ;Ergebnis 1 Byte nach rechts. B = E
0893  5A          LD    E,D          ;E = D
0894  51          LD    D,C          ;D = C
0895  4F          LD    C,A          ;C = 0
0896  C9          RET

                    ****************************************

                    Division mit einfacher Genauigkeit

                    X = X / 10
0897  CD A4 09      CALL  09A4H        ;Wert in X auf Stack retten
089A  21 D8 0D      LD    HL,0DD8H     ;Konstante 10 adressieren
089D  CD B1 09      CALL  09B1H        ;in X übertragen
08A0  C1            POP   BC           ;ehem. X-Wert in Y laden
08A1  D1            POP   DE

                    X = Y / X
08A2  CD 55 09      CALL  0955H        ;Divisor = 0 ?
08A5  CA 9A 19      JP    Z,199AH      ;ja, DIVISION BY ZERO - Fehler
08A8  2E FF         LD    L,0FFH       ;Flag Expon.verarb. für Division
08AA  CD 14 09      CALL  0914H        ;Exponenten und Vorzeichen verarb.
08AD  34            INC   (HL)         ;Exponenten-Ergebnis korrigieren
08AE  34            INC   (HL)         ;+ 2  (0914 = Exp.Y - Exp.X - 1)
08AF  2B            DEC   HL           ;HL auf MSB X
08B0  7E            LD    A,(HL)       ;X in Divisions-UP (ab 7840H)
08B1  32 89 78      LD    (7889H),A    ;MSB
08B4  2B            DEC   HL
```

```
08B5  7E          LD    A,(HL)        ;n. Byte
08B6  32 85 78    LD    (7885H),A
08B9  2B          DEC   HL
08BA  7E          LD    A,(HL)        ;LSB
08BB  32 81 78    LD    (7881H),A
08BE  41          LD    B,C           ;Y nach B,H,L übertragen (Divident)
08BF  EB          EX    DE,HL
08C0  AF          XOR   A             ;Y = 0 (für Quotient)
08C1  4F          LD    C,A
08C2  57          LD    D,A
08C3  5F          LD    E,A
08C4  32 8C 78    LD    (788CH),A     ;MSB Divisor = 0 (für Schieben)
08C7  E5          PUSH  HL            ;Divident auf Stack
08C8  C5          PUSH  BC
08C9  7D          LD    A,L           ;LSB Divident laden
08CA  CD 80 78    CALL  7880H         ;Divident - Divisor
08CD  DE 00       SBC   A,0           ;MSB Divident = übertrag,Unterlauf?
08CF  3F          CCF                 ;Carry komplementieren
08D0  30 07       JR    NC,08D9H      ;ja - Subtr. zurück, 0 in Quotient
08D2  32 8C 78    LD    (788CH),A     ;MSB Divident in UP
08D5  F1          POP   AF            ;Divident vom Stack entfernen
08D6  F1          POP   AF
08D7  37          SCF                 ;Carry-Flag setzen (1 in Quotient)
08D8  D2          DEFB  0D2H          ;JP NC,0E1C1H  wird nie ausgeführt.
                                      ;zum überspringen der folg. 2 POPs
08D9  D1          POP   BC            ;Divident vom Stack holen
08DA  E1          POP   HL            ;= Subtraktion rückgängig machen
08DB  79          LD    A,C           ;MSB d. Quotienten in A
08DC  3C          INC   A             ;Bit 7 testen
08DD  3D          DEC   A
08DE  1F          RRA                 ;letztes Bit für Rundung in Bit 7
08DF  FA 97 07    JP    M,0797H       ;war Bit 7 bei INC/DEC=1, ja-fertig
08E2  17          RLA                 ;Quotient 1 Bit nach links
08E3  7B          LD    A,E           ;Ergebnisbit (0 o. 1) wird
08E4  17          RLA                 ;aus dem Carry-Bit eingeschoben
08E5  5F          LD    E,A
08E6  7A          LD    A,D
08E7  17          RLA
08E8  57          LD    D,A
08E9  79          LD    A,C
08EA  17          RLA
08EB  4F          LD    C,A
08EC  29          ADD   HL,HL         ;Divident * 2
08ED  78          LD    A,B
```

```
08EE  17           RLA
08EF  47           LD    B,A
08F0  3A 8C 78     LD    A,(788CH)     ;= MSB Divident
08F3  17           RLA
08F4  32 8C 78     LD    (788CH),A
08F7  79           LD    A,C           ;ist das Ergebnis noch 0?
08F8  B2           OR    D
08F9  B3           OR    E
08FA  20 CB        JR    NZ,08C7H      ;nein - weiter
08FC  E5           PUSH  HL            ;Divident LSB retten
08FD  21 24 79     LD    HL,7924H      ;Quotient Exponent adressieren
0900  35           DEC   (HL)          ;- 1
0901  E1           POP   HL            ;Divident LSB wieder laden
0902  20 C3        JR    NZ,08C7H      ;Quotient Exp. ungleich 0 - weiter
0904  C3 B2 07     JP    07B2H         ;Exponent = 0, OVERFLOW - Error
```

```
                        ****************************************

                        Verarbeiten der Exponenten und Vorzeichen für
                        Multiplikation und Division

                        Einsprung: Division, doppelte Genauigkeit
0907  3E FF               LD      A,0FFH          ;Flag für Divison setzen
0909  2E                  DEFB    2EH             ;LD L,0AFH zum überspringen des XOR

                        Einsprung: Multiplikation, doppelte Genauigkeit
090A  AF                  XOR     A               ;Flag für Multiplikation setzen
090B  21 2D 79            LD      HL,792DH        ;MSB Y adressieren
090E  4E                  LD      C,(HL)          ;Vorzeichen Y in C
090F  23                  INC     HL              ;HL auf Exponent X
0910  AE                  XOR     (HL)            ;mit dem Flag verknüpfen
0911  47                  LD      B,A             ;Mult: B=Exp.Y  Div: B=-Exp.Y-1
0912  2E 00               LD      L,0             ;Flag in L = 0

                        Einsprung: Multiplikation, einfache Genauigkeit (L=0)
                                   Division, einfache Genauigkeit (L=FF)
0914  78                  LD      A,B             ;Exponent Y laden
0915  B7                  OR      A               ;= 0 ? (d.h. Y = 0)
0916  28 1F               JR      Z,0937H         ;ja, sofort ins Hauptprog. zurück
0918  7D                  LD      A,L             ;Flag laden
0919  21 24 79            LD      HL,7924H        ;Exp. X adressieren
091C  AE                  XOR     (HL)            ;mit Flag verknüpfen, d.h. bei
                                                  ;Division einf. Genauigk. -Exp.X-1
                                                  ;sonst unverändert.
091D  80                  ADD     A,B             ;+ Exponent Y
091E  47                  LD      B,A             ;Summe nach Exponent Y
091F  1F                  RRA                     ;über- oder Unterlauf?
0920  A8                  XOR     B
0921  78                  LD      A,B             ;neuen Exponent Y laden
0922  F2 36 09            JP      P,0936H         ;Unter-/Überlauf
0925  C6 80               ADD     A,80H           ;Offset addieren
0927  77                  LD      (HL),A          ;und als neuen Exponenten X speich.
0928  CA 90 08            JP      Z,0890H         ;= 0? ja-zum Hauptprogramm zurück
092B  CD DF 09            CALL    09DFH           ;Vorzeichen verarbeiten
092E  77                  LD      (HL),A          ;in Zwischenspeicher (7925H)
092F  2B                  DEC     HL              ;Exponent X adressieren
0930  C9                  RET

                        Exponenten über-/Unterlauf
0931  CD 55 09            CALL    0955H           ;Vorzeichen von X testen
```

```
0934  2F           CPL              ;Ergebnis komplementieren
0935  E1           POP   HL         ;Rücksprungadr. vom Stack entfernen
                                    ;direkt in Ausdrucksanalyse zurück
0936  B7           OR    A          ;war es ein Unterlauf?
0937  E1           POP   HL         ;noch eine Rsp-Adresse vom Stack
                                    ;d.h. sofort ins Hauptprogramm zur.
0938  F2 78 07      JP    P,0778H    ;Unterlauf, X=0, RET
093B  C3 B2 07      JP    07B2H      ;OVERFLOW-Error

                    ************************************

                    Multiplikation einfacher Genauigkeit mit 10

                    X = X * 10
093E  CD BF 09      CALL  09BFH      ;X in Y übertragen
0941  78            LD    A,B        ;Wert = 0 ? (Exp.Y=0)
0942  B7            OR    A
0943  C8            RET   Z          ;ja, fertig
0944  C6 02         ADD   A,2        ;Exp. Y + 2, d.h. Y = Wert * 4
0946  DA B2 07      JP    C,07B2H    ;bei überlauf OVERFLOW-Error
0949  47            LD    B,A        ;Exponent zurück in Y
094A  CD 16 07      CALL  0716H      ;X = X + Y, d.h. X = Wert * 5
094D  21 24 79      LD    HL,7924H   ;Exponent X + 1
0950  34            INC   (HL)       ;d.h. X = Wert * 10
0951  C0            RET   NZ         ;überlauf? nein-zurück
0952  C3 B2 07      JP    07B2H      ;ja - OVERFLOW-Error

                    ************************************

                    Test einer reellen Zahl
                    Eing.: X = Zahl (einfache o. doppelte Genauigkeit)
                    Ausg.: wenn X < 0, A=FF CY=1 S=1
                           wenn X = 0, A=00 Z=1  P=1
                           wenn X > 0, A=01
0955  3A 24 79      LD    A,(7924H)  ;Exponent X laden
0958  B7            OR    A          ;= 0? (X = 0)
0959  C8            RET   Z          ;ja!
095A  3A 23 79      LD    A,(7923H)  ;MSB X laden
095D  FE            DEFB  0FEH       ;CP 2F - Dummy-Befehl, elim. CPL
095E  2F            CPL              ;A komplemtieren (sep.Einsprung)
095F  17            RLA              ;Vorzeichen X in Carry schieben
0960  9F            SBC   A,A        ;A = 0 - Carry
0961  C0            RET   NZ         ;X > 0? nein - fertig
0962  3C            INC   A          ;ja,  A = 1 setzen
```

```
0963  C9              RET

                      ************************************

                      8-Bit-Zahl mit Vorzeichen in Zahl einfacher
                      Genauigkeit umwandeln
                      Eing.: A = Zahl
                      Ausg.: X = Zahl in einf.Genauigkeit
0964  06 88           LD    B,88H         ;Exponent des Ergebnisses in B
0966  11 00 00        LD    DE,0          ;für Normalisierung löschen
0969  21 24 79        LD    HL,7924H      ;Adresse Exponent in X
096C  4F              LD    C,A           ;umzuwandelnde Zahl in C
096D  70              LD    (HL),B        ;Exponent nach X übertragen
096E  06 00           LD    B,0           ;B=0 für Normalisierung
0970  23              INC   HL            ;Adresse Vorzeichenbyte in X (MSB)
0971  36 80           LD    (HL),80H      ;Vorzeichen = "-" setzen
0973  17              RLA                 ;Vorz. der umzuw. Zahl ins Carry
0974  C3 62 07        JP    0762H         ;zur Normalisierung

                      ************************************

                      ABS - Funktion
                      Absolutwert einer Zahl bilden
                      Eing.: X = Zahl
                      Ausg.: X = Absolutwert der Zahl
0977  CD 94 09        CALL  0994H         ;X > 0 ?
097A  F8              RET   P             ;ja, fertig

                      Zahl in X invertieren
097B  E7              RST   20H           ;Typ X prüfen
097C  FA 5B 0C        JP    M,0C5BH       ;Integer? ja - weiter bei C5B
097F  CA F6 0A        JP    Z,0AF6H       ;String? ja - TYPE MISMATCH Error

                      Reele Zahl in X invertieren
0982  21 23 79        LD    HL,7923H      ;MSB X adressieren
0985  7E              LD    A,(HL)        ;und laden
0986  EE 80           XOR   80H           ;Vorzeichen invertieren
0988  77              LD    (HL),A        ;MSB X zurückschreiben
0989  C9              RET

                      ************************************

                      SGN - Funktion
                      Eing.: X = Zahl
```

```
                        Ausg.: X = 0, wenn Zahl = 0
                               X = 1, wenn Zahl positiv
                               X = -1, wenn Zahl negativ
098A  CD 94 09          CALL   0994H           ;X testen

                        A in 16-Bit Integer umwandeln (mit Vorzeichen)
098D  6F                LD     L,A             ;Zahl in L
098E  17                RLA                    ;Zahl < 0?
098F  9F                SBC    A,A             ;ja, -1 in A und H
0990  67                LD     H,A             ;nein, 0 in A und H
0991  C3 9A 0A          JP     0A9AH           ;HL nach X übertragen

                        Alle numerischen Typen testen
0994  E7                RST    20H             ;Typ prüfen
0995  CA F6 0A          JP     Z,0AF6H         ;String? ja - TYPE MISMATCH Error
0998  F2 55 09          JP     P,0955H         ;Einf. o. doppelte Genauigkeit

                        Integer - Zahl testen
099B  2A 21 79          LD     HL,(7921H)      ;Integer-Zahl in HL
099E  7C                LD     A,H             ;= 0 ?
099F  B5                OR     L
09A0  C8                RET    Z               ;ja - fertig
09A1  7C                LD     A,H             ;nein - MSB in A
09A2  18 BB             JR     095FH           ;weiter bei 095FH

                        ************************************

                        Zahlen verschiedenen Typs transportieren

                        von X auf Stack (einfache Genauigkeit)
09A4  EB                EX     DE,HL           ;HL in DE sichern
09A5  2A 21 79          LD     HL,(7921H)      ;LSB X in HL
09A8  E3                EX     (SP),HL         ;mit RET-Adresse auf Stack tauschen
09A9  E5                PUSH   HL              ;RET-Adresse wieder auf Stack
09AA  2A 23 79          LD     HL,(7923H)      ;MSB X + Exp. X in HL
09AD  E3                EX     (SP),HL         ;mit RET-Adresse auf Stack tauschen
09AE  E5                PUSH   HL              ;RET-Adresse wieder auf Stack
09AF  EB                EX     DE,HL           ;Inhalt von HL wiederherstellen
09B0  C9                RET

                        Zahl einfacher Genauigkeit von RAM in X
                        Eing.: HL = Adresse der Zahl im Speicher
09B1  CD C2 09          CALL   09C2            ;Zahl nach Y übertragen
```

```
                        Zahl einfacher Genauigkeit von Y nach X
09B4  EB                EX      DE,HL           ;LSB Y in HL, HL in DE sichern
09B5  22 21 79          LD      (7921H),HL      ;HL in LSB Y übertragen
09B8  60                LD      H,B             ;MSB Y + Exp. Y in HL
09B9  69                LD      L,C
09BA  22 23 79          LD      (7923H),HL      ;als MSB X u. Exp. X abspeichern
09BD  EB                EX      DE,HL           ;Inhalt von HL wiederherstellen
09BE  C9                RET

                        Zahl einfacher Genauigkeit von X nach Y
09BF  21 21 79          LD      HL,7921H        ;LSB X adressieren
09C2  5E                LD      E,(HL)          ;LSB laden
09C3  23                INC     HL              ;nächstes Byte
09C4  56                LD      D,(HL)          ;laden
09C5  23                INC     HL              ;MSB laden
09C6  4E                LD      C,(HL)
09C7  23                INC     HL              ;Exp. laden
09C8  46                LD      B,(HL)
09C9  23                INC     HL              ;HL hinter die Zahl
09CA  C9                RET

                        Zahl einfacher Genauigkeit von X in RAM übertragen
09CB  11 21 79          LD      DE,7921H        ;X-Adresse in DE
09CE  06 04             LD      B,4             ;Anzahl Bytes f. einf. Genauigkeit
09D0  18 05             JR      09D7H

                        Zahl jeden Typs von (HL) nach (DE) transportieren
09D2  EB                EX      DE,HL           ;Ziel- und Quelladresse vertauschen

                        Zahl jeden Typs von (DE) nach (HL) transportieren
09D3  3A AF 78          LD      A,(78AFH)       ;Typ der Zahl laden
09D6  47                LD      B,A             ;dient als Bytezähler
09D7  1A                LD      A,(DE)          ;Byte laden
09D8  77                LD      (HL),A          ;und in neuen Bereich übertragen
09D9  13                INC     DE              ;Adressen + 1
09DA  23                INC     HL
09DB  05                DEC     B               ;Zähler - 1
09DC  20 F9             JR      NZ,09D7H        ;> 0 ? ja - zurück
09DE  C9                RET                     ;fertig

                        Verarbeitung von Vorzeichen bei reellen Zahlen
09DF  21 23 79          LD      HL,7923H        ;MSB X adressieren
09E2  7E                LD      A,(HL)          ;und in A laden
09E3  07                RLCA                    ;Vorzeichen in Bit 0 von A
```

```
09E4  37           SCF                 ;Carry = 1 setzen
09E5  1F           RRA                 ;Vorzeichen in Carry, MSB X(7) = 1
09E6  77           LD    (HL),A        ;in MSB X zurück
09E7  3F           CCF                 ;Vorzeichen komplementieren
09E8  1F           RRA                 ;und in A(7)
09E9  23           INC   HL            ;Adresse HL auf Vorzeichen-Flag
09EA  23           INC   HL            ;(7925H)
09EB  77           LD    (HL),A        ;komplement. Vorzeichen ablegen
09EC  79           LD    A,C           ;MSB Y in A
09ED  07           RLCA                ;Vorzeichen Y in Bit 0 von A
09EE  37           SCF                 ;Carry = 1 setzen
09EF  1F           RRA                 ;MSB Y(7) = 1, Vorz. Y in Carry
09F0  4F           LD    C,A           ;MSB Y zurück
09F1  1F           RRA                 ;Vorzeichen in A(7)
09F2  AE           XOR   (HL)          ;mit kompl. Vorz.X verknüpfen
                                       ;A(7) = 1, wenn Vorz. X = Vorz. Y
09F3  C9           RET

                   Wert jeden Typs von Y nach X transportieren
                   (78AFH = Typ des Wertes)
09F4  21 27 79     LD    HL,7927H      ;Y-Adresse in HL
09F7  11 D2 09     LD    DE,09D2H      ;Adresse der Transport-Routine
09FA  18 06        JR    0A02H

                   Wert jeden Typs von X nach Y transportieren
                   (78AFH = Typ des Wertes)
09FC  21 27 79     LD    HL,7927H      ;Y-Adresse in HL
09FF  11 D3 09     LD    DE,09D3H      ;Adresse der Transport-Routine
0A02  D5           PUSH  DE            ;Adr. Transport-Routine auf Stack

                   X-Adresse in Abhängigkeit vom Typ ermitteln
0A03  11 21 79     LD    DE,7921H      ;X-Adr. f. Integer, Strings und
                                       ;einfache Genauigkeit
0A06  E7           RST   20H           ;Typ testen
0A07  D8           RET   C             ;doppelte Genauigkeit? nein-fertig
0A08  11 1D 79     LD    DE,791DH      ;X-Adr. f. doppelte Genauigkeit
0A0B  C9           RET

                   ************************************

                   Vergleichs-Routinen

                   Vergleich von Zahlen einfacher Genauigkeit
                   Eing.: X = Zahl 1
```

```
                        Y = Zahl 2
                Ausg.:  X > Y, A = 1
                        X = Y, A = 0, Z = 1
                        X < Y, A = FF, CY = 1, S = 1
0A0C  78                LD      A,B             ;Y = 0?
0A0D  B7                OR      A
0A0E  CA 55 09          JP      Z,0955H         ;ja - X testen und zurück
0A11  21 5E 09          LD      HL,095EH        ;Adr. der Testroutine auf Stack
0A14  E5                PUSH    HL
0A15  CD 55 09          CALL    0955H           ;X = 0?
0A18  79                LD      A,C             ;MSB Y in A
0A19  C8                RET     Z               ;ja,  -Vorzeichen von Y = Ergebnis
0A1A  21 23 79          LD      HL,7923H        ;Adresse von MSB X laden
0A1D  AE                XOR     (HL)            ;Vorzeichen X = Vorzeichen Y ?
0A1E  79                LD      A,C             ;MSB Y in A
0A1F  F8                RET     M               ;nein, -Vorzeichen von Y = Ergebnis
0A20  CD 26 0A          CALL    0A26H           ;Vergleich bei gleichen Vorzeichen
0A23  1F                RRA                     ;Carry in Bit 7 von A
0A24  A9                XOR     C               ;wenn Y negativ, A(7) invertieren
0A25  C9                RET

0A26  23                INC     HL              ;Adresse des Exp. X in HL
0A27  78                LD      A,B             ;Exp. Y laden
0A28  BE                CP      (HL)            ;Vergleich der beiden Exponenten
0A29  C0                RET     NZ              ;zurück, wenn ungleich
0A2A  2B                DEC     HL              ;MSB X mit MSB Y vergleichen
0A2B  79                LD      A,C
0A2C  BE                CP      (HL)
0A2D  C0                RET     NZ              ;zurück, wenn ungleich
0A2E  2B                DEC     HL              ;n.Byte X mit n.Byte Y
0A2F  7A                LD      A,D
0A30  BE                CP      (HL)
0A31  C0                RET     NZ              ;zurück, wenn ungleich
0A32  2B                DEC     HL              ;LSB X mit LSB Y vergleichen
0A33  7B                LD      A,E
0A34  96                SUB     (HL)
0A35  C0                RET     NZ              ;zurück, wenn ungleich
0A36  E1                POP     HL              ;X = Y, RET-Adressen vom Stack entf
0A37  E1                POP     HL
0A38  C9                RET                     ;zurück ins Hauptprogramm mit Z=1

                Integer-Vergleich
                Eing.:  HL = Zahl1 (Z1)
                        DE = Zahl2 (Z2)
```

```
                        Ausg.: Z1 > Z2, A = 1
                               Z1 = Z2, A = 0, Z = 1
                               Z1 < Z2, A = FF, CY = 1, S = 1
0A39  7A                LD     A,D             ;Vorzeichen gleich ?
0A3A  AC                XOR    H
0A3B  7C                LD     A,H             ;MSB Z1 in A
0A3C  FA 5F 0A          JP     M,095FH         ;nein, Vorzeichen von Z1 = Ergebnis
0A3F  BA                CP     0               ;MSB Z1 = MSB Z2 ?
0A40  C2 60 09          JP     NZ,0960H        ;nein, Carry ergibt Ergebnis
0A43  7D                LD     A,L             ;LSB Z1 = LSB Z2 ?
0A44  93                SUB    E
0A45  C2 60 09          JP     NZ,0960H        ;nein, Carry ergibt Ergebnis
0A48  C9                RET

                        Vergleich doppelter Genauigkeit

                        X mit Konstante (DE)
0A49  21 27 79          LD     HL,7927H        ;Adresse Y laden
0A4C  CD D3 09          CALL   09D3H           ;Konstante in Y übertragen

                        X mit Y vergleichen (Y=7927 ff.)
                        Ausg.: X > Y, A = 1
                               X = Y, A = 0, Z = 1
                               X < Y, A = FF, CY = 1, S = 1
0A4F  11 2E 79          LD     DE,792EH        ;Exponent Y adressieren
0A52  1A                LD     A,(DE)          ;Y = 0 ?
0A53  B7                OR     A
0A54  CA 55 09          JP     Z,0955H         ;ja, X bestimmt das Ergebnis
0A57  21 5E 09          LD     HL,095EH        ;Adresse der Testroutine auf Stack
0A5A  E5                PUSH   HL
0A5B  CD 55 09          CALL   0955H           ;X = 0 ?
0A5E  1B                DEC    DE              ;Adresse MSB Y
0A5F  1A                LD     A,(DE)          ;MSB Y in A und C
0A60  4F                LD     C,A
0A61  C8                RET    Z               ;X = 0, -Vorzeichen Y = Ergebnis
0A62  21 23 79          LD     HL,7923H        ;Adresse MSB X
0A65  AE                XOR    (HL)            ;Vorzeichen X = Vorzeichen Y ?
0A66  79                LD     A,C
0A67  F8                RET    M               ;nein, -Vorzeichen Y = Ergebnis
0A68  13                INC    DE              ;Exponent Y adressieren
0A69  23                INC    HL              ;Exponent X adressieren
0A6A  06 08             LD     B,8             ;8 Bytes vergleichen
0A6C  1A                LD     A,(DE)          ;Byte von Y laden
0A6D  96                SUB    (HL)            ;- X-Byte
```

```
0A6E  C2 23 0A        JP    NZ,0A23H      ;ungleich, aus Carry Ergebnis erm.
0A71  1B              DEC   DE            ;Adresspointer X,Y -1
0A72  2B              DEC   HL
0A73  05              DEC   B             ;8 Bytes verglichen?
0A74  20 F6           JR    NZ,0A6CH      ;nein, nächstes Byte
0A76  C1              POP   BC            ;Rücksprungadresse vom Stack entf.
0A77  C9              RET                 ;mit A=0 und Z=1 zurück

                      Y mit X vergleichen
0A78  CD 4F 0A        CALL  0A4FH         ;o.a. Vergleichsroutine aufrufen
0A7B  C2 5E 09        JP    NZ,095EH      ;wenn ungleich, Ergebnis invertier.
0A7E  C9              RET

                      ************************************

                      CINT - Funktion
                      Zahl in 16-Bit Integer umwandeln
                      Eing.: X=Ausgangswert
                      Ausg.: X=Integer
0A7F  E7              RST   20H           ;Typ des Ausgangswertes testen
0A80  2A 21 79        LD    HL,(7921H)    ;X-Adresse in HL
0A83  F8              RET   M             ;Integer? ja - fertig!
0A84  CA F6 0A        JP    Z,0AF6H       ;String? ja - TYPE MISMATCH Error
0A87  D4 B9 0A        CALL  NC,0AB9H      ;Dopp.Genauigk.? ja - zunächst in
                                          ;einfache Genauigkeit umwandeln
0A8A  21 B2 07        LD    HL,07B2H      ;OVERFLOW Error - Adresse in Stack
0A8D  E5              PUSH  HL
0A8E  3A 24 79        LD    A,(7924H)     ;Abs. X > 32767 ? (Exp.X > 16)
0A91  FE 90           CP    90H
0A93  30 0E           JR    NC,0AA3H      ;ja!
0A95  CD FB 0A        CALL  0AFBH         ;Integer X in DE
0A98  EB              EX    DE,HL         ;weiter in HL
0A99  D1              POP   DE            ;OV-Error-Adresse aus Stack entf.
0A9A  22 21 79        LD    (7921H),HL    ;HL in X übertragen

0A9D  3E 02           LD    A,2           ;Typ = Integer
0A9F  32 AF 78        LD    (78AFH),A     ;setzen
0AA2  C9              RET

0AA3  01 80 90        LD    BC,9080H      ;- 32768 in Y (BCDE)
0AA6  11 00 00        LD    DE,0
0AA9  CD 0C 0A        CALL  0A0CH         ;X = -32768 ?
0AAC  C0              RET   NZ            ;nein, OVERFLOW - Error
0AAD  61              LD    H,C           ;ja, HL = -32768
```

```
0AAE  6A          LD    L,D
0AAF  18 E8       JR    0A99H          ;weiter bei 0A99H

                  ****************************************

                  CSNG - Funktion
                  Zahl in Wert einfacher Genauigkeit umwandeln
                  Eing.: X = Ausgangswert
                  Ausg.: X = Wert in einfacher Genauigkeit
0AB1  E7          RST   20H            ;Typ des Ausgangswerts ermitteln
0AB2  E0          RET   PO             ;Ist schon einfache Genauigkeit!
0AB3  FA CC 0A    JP    M,0ACCH        ;Integer? ja - weiter bei 0ACCH
0AB6  CA F6 0A    JP    Z,0AF6H        ;String? ja - TYPE MISMATCH Error
0AB9  CD BF 09    CALL  09BFH          ;X nach Y übertragen
0ABC  CD EF 0A    CALL  0AEFH          ;Typ = einf. Genauigkeit
0ABF  78          LD    A,B            ;X = 0?
0AC0  B7          OR    A
0AC1  C8          RET   Z              ;ja, fertig
0AC2  CD DF 09    CALL  09DFH          ;nein, Vorzeichen abtrennen
0AC5  21 20 79    LD    HL,7920H       ;Erstes, nicht zu übernehm. Byte
0AC8  46          LD    B,(HL)         ;zur Rundung bereitstellen
0AC9  C3 96 07    JP    0796H          ;Runden und Normalisieren

0ACC  2A 21 79    LD    HL,(7921H)     ;Integer in HL
0ACF  CD EF 0A    CALL  0AEFH          ;Typ = einfache Genauigkeit
0AD2  7C          LD    A,H            ;Parameter für Umwandlung
0AD3  55          LD    D,L            ;bereitstellen
0AD4  1E 00       LD    E,0
0AD6  06 90       LD    B,90H          ;Exponent = 16 setzen
0AD8  C3 69 09    JP    0969H          ;zur Umwandlungsroutine

                  ****************************************

                  CDBL - Funktion
                  Zahl in Wert doppelter Genauigkeit umwandeln
                  Eing.: X = Ausgangszahl
                  Ausg.: X = Wert in doppelter Genauigkeit
0ADB  E7          RST   20H            ;Typ der Zahl ermitteln
0ADC  D0          RET   NC             ;ist schon doppelte Genauigkeit
0ADD  CA F6 0A    JP    Z,0AF6H        ;String? TYPE MISMATCH Error
0AE0  FC CC 0A    CALL  M,0ACCH        ;Integer? zuerst in einf.Genauigk.
0AE3  21 00 00    LD    HL,0           ;die vier niederwertigen Bytes
0AE6  22 1D 79    LD    (791DH),HL     ;in X löschen
0AE9  22 1F 79    LD    (791FH),HL
```

```
                        Typ = doppelte Genauigkeit
0AEC  3E 08             LD      A,8             ;A = Typcode 8
0AEE  01                DEFB    1               ;LD BC,043E = Dummy-Befehl
                                                ;weiter bei 0AF1H

                        Typ = einfache Genauigkeit
0AEF  3E 04             LD      A,4             ;A = Typcode 4
0AF1  C3 9F 0A          JP      0A9FH           ;in Typ-Byte (78AF) speichern

                        Prüfen, ob X einen String enthält
0AF4  E7                RST     20H             ;Typ-Byte auswerten
0AF5  C8                RET     Z               ;String? ja, fertig

0AF6  1E 18             LD      E,18H           ;Fehlercode f. TYPE MISMATCH Error
0AF8  C3 A2 19          JP      19A2H           ;zur Fehlerausgabe-Routine

                        ************************************

                        Gemeinsames Unterprogramm für INT, FIX, CINT

0AFB  47                LD      B,A             ;wenn A = 0, zurück mit Y = 0
0AFC  4F                LD      C,A
0AFD  57                LD      D,A
0AFE  5F                LD      E,A
0AFF  B7                OR      A
0B00  C8                RET     Z               ;ok, = 0
0B01  E5                PUSH    HL              ;Adresse von Exp. X retten
0B02  CD BF 09          CALL    09BFH           ;X nach Y übertragen
0B05  CD DF 09          CALL    09DFH           ;Vorzeichen abtrennen
0B08  AE                XOR     (HL)            ;Ist X negativ?
0B09  67                LD      H,A             ;Vorzeichen in H(7)
0B0A  FC 1F 0B          CALL    M,0B1FH         ;X = neg., LSB X - 1
0B0D  3E 98             LD      A,98H           ;Mantissenlänge - Exponent
0B0F  90                SUB     B               ;= Anzahl Rechtsverschiebungen
0B10  CD D7 07          CALL    07D7H           ;Rechtsverschiebungen durchführen
0B13  7C                LD      A,H             ;war X negativ ?
0B14  17                RLA
0B15  DC A8 07          CALL    C,07A8H         ;ja, Festkommazahl + 1
0B18  06 00             LD      B,0             ;LSB = 0
0B1A  DC C3 07          CALL    C,07C3H         ;wenn negativ, X = -X
0B1D  E1                POP     HL              ;Exponenten-Adresse laden
0B1E  C9                RET
```

```
0B1F  1B          DEC   DE          ;LSB - 1
0B20  7A          LD    A,D         ;= 0 ?
0B21  A3          AND   E
0B22  3C          INC   A
0B23  C0          RET   NZ          ;nein, fertig!
0B24  0B          DEC   BC          ;MSB - 1
0B25  C9          RET

                  ************************************

                  FIX - Funktion
                  Integer ohne Berücksichtigung des Vorzeichens bilden
                  Eing.: X = Ausgangswert
                  Ausg.: X = Funktionswert
0B26  E7          RST   20H         ;Typ ermitteln
0B27  F8          RET   M           ;bereits Integer? ja - fertig
0B28  CD 55 09    CALL  0955H       ;X >= 0?
0B2B  F2 37 0B    JP    P,0B37H     ;ja, weiter bei 0B37H
0B2E  CD 82 09    CALL  0982H       ;nein, X = -X
0B31  CD 37 0B    CALL  0B37H       ;Integer bilden
0B34  C3 7B 09    JP    097BH       ;X = -X

                  ************************************

                  INT - Funktion
                  Ermittelt nächst niedrigere ganze Zahl
                  Eing.: X = Ausgangswert
                  Ausg.: X = Integer
0B37  E7          RST   20H         ;Typ ermitteln
0B38  F8          RET   M           ;bereits Integer? ja - fertig
0B39  30 1E       JR    NC,0B59H    ;doppelte Genauigkeit? ja - Sprung
0B3B  28 B9       JR    Z,0AF6H     ;String? ja - TYPE MISMATCH Error
0B3D  CD 8E 0A    CALL  0A8EH       ;wenn möglich, in Integer umwandeln
0B40  21 24 79    LD    HL,7924H    ;Exponent X adressieren
0B43  7E          LD    A,(HL)      ;und in A laden
0B44  FE 98       CP    98H         ;Exponent >= Mantissenlänge?
0B46  3A 21 79    LD    A,(7921H)   ;LSB von X laden
0B49  D0          RET   NC          ;ja, fertig - keine Nachkommastell.
0B4A  7E          LD    A,(HL)      ;Exponent X laden
0B4B  CD FB 0A    CALL  0AFBH       ;Nachkommastellen entfernen
                                    ;und in Y übertragen
0B4E  36 98       LD    (HL),98H    ;Mantissenlänge in Exp. X eintragen
0B50  7B          LD    A,E         ;LSB Y in A
0B51  F5          PUSH  AF          ;auf Stack sichern
```

```
0B52  79           LD    A,C          ;Vorzeichen Y in Carry
0B53  17           RLA
0B54  CD 62 07     CALL  0762H        ;Normalisieren, wenn Y<0, X=-X
0B57  F1           POP   AF           ;LSB Y wieder laden
0B58  C9           RET

                   Doppelte Genauigkeit in Integer umwandeln
0B59  21 24 79     LD    HL,7924H     ;Adresse des Exponenten laden
0B5C  7E           LD    A,(HL)       ;Exponent < 16? (X < 32768)
0B5D  FE 90        CP    90H
0B5F  DA 7F 0A     JP    C,0A7FH      ;ja, weiter bei der CINT-Funktion
0B62  20 14        JR    NZ,0B78H     ;Exponent > 16, weiter bei 0B78H
0B64  4F           LD    C,A          ;Exponent in C
0B65  2B           DEC   HL           ;Adresse auf MSB
0B66  7E           LD    A,(HL)       ;X = -32768 ?
0B67  EE E0        XOR   80H          ;-Vorzeichen von X in A(7)
0B69  06 06        LD    B,6          ;Bytezähler = 6
0B6B  2B           DEC   HL           ;nächstes Byte
0B6C  B6           OR    (HL)         ;wenn ungleich 0, dann nicht.
0B6D  05           DEC   B            ;Zähler -1
0B6E  20 FB        JR    NZ,0B6BH     ;wenn 0, fertig
0B70  B7           OR    A            ;A = 0? (d.h. X=-32768?)
0B71  21 00 80     LD    HL,8000H     ;-32768 in HL
0B74  CA 9A 0A     JP    Z,0A9AH      ;ja, fertig!
0B77  79           LD    A,C          ;Exponent X zurück in A
0B78  FE B8        CP    0B8H         ;Exponent >= Mantissenlänge?
0B7A  D0           RET   NC           ;ja, fertig - keine Kommastellen
0B7B  F5           PUSH  AF           ;Flags retten
0B7C  CD BF 09     CALL  09BFH        ;X nach Y übertragen
0B7F  CD DF 09     CALL  09DFH        ;Vorzeichen abtrennen
0B82  AE           XOR   (HL)         ;X = negativ ?
0B83  2B           DEC   HL           ;Adresse Exponent in HL
0B84  36 B8        LD    (HL),0B8H    ;Exponent X = Mantissenlänge
0B86  F5           PUSH  AF           ;Vorzeichen A(7) retten
0B87  FC A0 0B     CALL  M,0BA0H      ;X negativ, LSB -1
0B8A  21 23 79     LD    HL,7923H     ;HL = Adresse MSB X
0B8D  3E B8        LD    A,0B8H       ;Mantissenlänge - Exponent
0B8F  90           SUB   B            ;= Anzahl Rechtsverschiebungen
0B90  CD 69 0D     CALL  0D69H        ;rechts verschieben
0B93  F1           POP   AF           ;Vorzeichen zurück
0B94  FC 20 0D     CALL  M,0D20H      ;wenn X < 0, Integer + 1
0B97  AF           XOR   A            ;LSB für Normalisierung = 0
0B98  32 1C 79     LD    (791CH),A
0B9B  F1           POP   AF           ;Flag für Normalisierung laden
```

```
0B9C  D0          RET   NC            ;keine Normalisierung - fertig
0B9D  C3 D8 0C    JP    0CD8H         ;Sprung zur Normalisierung
0BA0  21 1D 79    LD    HL,791DH      ;Adresse LSB X
0BA3  7E          LD    A,(HL)        ;LSB X laden
0BA4  35          DEC   HL            ;- 1
0BA5  B7          OR    A             ;war vorher 0?
0BA6  23          INC   HL            ;nächstes Byte in X
0BA7  28 FA       JR    Z,0BA3H       ;ja, weiter
0BA9  C9          RET

                  ************************************

                  Multiplikation (für Matrix-Verwaltung)
                  Eing.: BC = Faktor
                         DE = Faktor
                  Ausg.: DE = Produkt
0BAA  E5          PUSH  HL            ;HL retten
0BAB  21 00 00    LD    HL,0          ;Ergebnis = 0 setzen
0BAE  78          LD    A,B           ;Faktor = 0?
0BAF  B1          OR    C
0BB0  28 12       JR    Z,0BC4H       ;ja, Ergebnis = 0
0BB2  3E 10       LD    A,16          ;Zähler für 16 Durchläufe
0BB4  29          ADD   HL,HL         ;Ergebnis * 2, Überlauf ?
0BB5  DA 3D 27    JP    C,273DH       ;ja, BAD SUBSCRIPT - Fehler
0BB8  EB          EX    DE,HL         ;Faktor in DE * 2
0BB9  29          ADD   HL,HL
0BBA  EB          EX    DE,HL
0BBB  30 04       JR    NC,0BC1H      ;kein Überlauf, weiter
0BBD  09          ADD   HL,BC         ;ja, Faktor in BC addieren
0BBE  DA 3D 27    JP    C,273DH       ;Überlauf, BAD SUBSCRIPT - Fehler
0BC1  3D          DEC   A             ;Zähler - 1
0BC2  20 F0       JR    NZ,0BB4H      ;nicht 0, neuer Durchlauf
0BC4  EB          EX    DE,HL         ;Ergebnis in DE
0BC5  E1          POP   HL            ;HL wiederherstellen
0BC6  C9          RET

                  ************************************

                  Integer-Subtraktion
                  Eing.: DE = Minuend
                         HL = Subtrahend
                  Ausg.: HL = Differenz
                         (bei Unter-/Überlauf in X mit einf.Genauigkeit)
0BC7  7C          LD    A,H           ;Vorzeichen Subtrahend in Carry
```

```
0BC8  17           RLA
0BC9  9F           SBC   A,A          ;B=-1, wenn Subtrahend < 0, sonst 0
0BCA  47           LD    B,A
0BCB  CD 51 0C     CALL  0C51H        ;Subtrahend komplementieren
0BCE  79           LD    A,C          ;A = 0
0BCF  98           SBC   A,B          ;Vorzeichen-Flag komplementieren
0BD0  18 03        JR    0BD5H        ;Sprung zur Addition

                   ************************************

                   Integer-Addition
                   Eing.: DE = Summand
                          HL = Summand
                   Ausg.: HL = Summe
                          (bei Unter-/Überlauf in X mit einf. Genauigkeit)
0BD2  7C           LD    A,H          ;Vorz. des 2. Summanden in Carry
0BD3  17           RLA
0BD4  9F           SBC   A,A          ;Vorzeichen-Flag in B
0BD5  47           LD    B,A          ;B=-1, wenn Summ.< 0, sonst B=0
0BD6  E5           PUSH  HL           ;2. Summanden auf Stack
0BD7  7A           LD    A,D          ;Vorz. des 1. Summanden in Carry
0BD8  17           RLA
0BD9  9F           SBC   A,A          ;A=-1, wenn Summ.< 0, sonst A=0
0BDA  19           ADD   HL,DE        ;Summe bilden
0BDB  88           ADC   A,B          ;Überlauf? (wenn beide negativ und
                                      ;Ergebnis positiv, oder wenn beide
0BDC  0F           RRCA               ;positiv und das Ergebnis negativ.
0BDD  AC           XOR   H
0BDE  F2 99 0A     JP    P,0A99H      ;kein Überlauf, HL in X und fertig

                   Additions - Überlauf
0BE1  C5           PUSH  BC           ;Vorzeichen-Flag retten
0BE2  EB           EX    DE,HL        ;1. Summand in HL
0BE3  CD CF 0A     CALL  0ACFH        ;mit einf. Genauigkeit in X
0BE6  F1           POP   AF           ;Vorzeichen-Flag in A
0BE7  E1           POP   HL           ;2. Summand wieder laden
0BE8  CD A4 09     CALL  09A4H        ;X auf Stack schieben
0BEB  EB           EX    DE,HL        ;2. Summand in DE
0BEC  CD 6B 0C     CALL  0C6BH        ;und mit einf. Genauigkeit in X
0BEF  C3 8F 0F     JP    0F8FH        ;X + Summand vom Stack in X

                   ************************************

                   Integer - Multiplikation
```

```
                        Eing.: DE = Faktor
                               HL = Faktor
                        Ausg.: HL = Produkt
                               (bei Unter-/Überlauf mit einf. Genauigkeit in X)
0BF2  7C                LD     A,H             ;2. Faktor = 0 ?
0BF3  B5                OR     L
0BF4  CA 9A 0A          JP     Z,0A9AH         ;ja, Ergebnis = 0, fertig
0BF7  E5                PUSH   HL              ;2. Faktor retten
0BF8  D5                PUSH   DE              ;1. Faktor retten
0BF9  CD 45 0C          CALL   0C45H           ;Vorzeichen entfernen
                                               ;XOR der beiden Vorzeichen in B(7)
0BFC  C5                PUSH   BC              ;Vorzeichen-Flag retten
0BFD  44                LD     B,H             ;2. Faktor in BC
0BFE  4D                LD     C,L
0BFF  21 00 00          LD     HL,0            ;Ergebnis = 0 setzen
0C02  3E 10             LD     A,16            ;Zähler = 16 Durchläufe
0C04  29                ADD    HL,HL           ;Ergebnis * 2
0C05  38 1F             JR     C,0C26H         ;bei Überlauf, Sonderroutine
0C07  EB                EX     DE,HL           ;1. Faktor * 2
0C08  29                ADD    HL,HL
0C09  EB                EX     DE,HL           ;Überlauf?
0C0A  30 04             JR     NC,0C10H        ;nein, keine Addition
0C0C  09                ADD    HL,BC           ;ja, Ergebnis + 2. Faktor
0C0D  DA 26 0C          JP     C,0C26H         ;bei Überlauf Sonderroutine
0C10  3D                DEC    A               ;Zähler - 1
0C11  20 F1             JR     NZ,0C04H        ;nicht 0, nächster Durchlauf
0C13  C1                POP    BC              ;Vorzeichen-Flag laden
0C14  D1                POP    DE              ;1. Faktor vom Stack holen
0C15  7C                LD     A,H             ;Ergebnis > 32767 ?
0C16  B7                OR     A
0C17  FA 1F 0C          JP     M,0C1FH         ;ja, Überlauf!
0C1A  D1                POP    DE              ;2. Faktor vom Stack holen
0C1B  78                LD     A,B             ;Ergebnis mit Vorzeichen-Flag
0C1C  C3 4D 0C          JP     0C4DH           ;korrigieren

                        Multiplikations - Überlauf
0C1F  EE 80             XOR    80H             ;Ergebnis = 32768 ?
0C21  B5                OR     L
0C22  28 13             JR     Z,0C37H         ;ja!
0C24  EB                EX     DE,HL           ;1. Faktor in HL
0C25  01                DEFB   01H             ;LD BC,E1C1 = Dummy-Befehl
0C26  C1                POP    BC              ;Vorzeichen-Flag laden
0C27  E1                POP    HL              ;1. Faktor in HL laden
0C28  CD CF 0A          CALL   0ACFH           ;1. Faktor mit einf. Genauigk. in X
```

```
0C2B  E1           POP   HL          ;2. Faktor vom Stack holen
0C2C  CD A4 09     CALL  09A4H       ;1. Faktor aus X auf Stack
0C2F  CD CF 0A     CALL  0ACFH       ;2. Faktor mit einf. Genauigk. in X
0C32  C1           POP   BC          ;1. Faktor aus Stack in Y
0C33  D1           POP   DE
0C34  C3 47 08     JP    0847H       ;X = Y * X

0C37  78           LD    A,B         ;Vorzeichen-Flag in A
0C38  B7           OR    A           ;Ergebnis sollte negativ sein
0C39  C1           POP   BC          ;Stack bereinigen
0C3A  FA 9A 0A     JP    M,0A9AH     ;ist negativ, HL (-32768) in X
0C3D  D5           PUSH  DE          ;1. Faktor auf Stack
0C3E  CD CF 0A     CALL  0ACFH       ;HL (-32768) mit einf.Genauigk.in X
0C41  D1           POP   DE          ;1. Faktor wieder laden
0C42  C3 82 09     JP    0982H       ;X komplementieren, fertig

                   Vorzeichen verknüpfen,
                   bei negativen Faktoren Komplement bilden
0C45  7C           LD    A,H         ;wenn Vorzeichen gleich,
0C46  AA           XOR   D           ;B(7) = 0, bei ungleich B(7) = 1
0C47  47           LD    B,A
0C48  CD 4C 0C     CALL  0C4CH       ;Absolutwerte bilden
0C4B  EB           EX    DE,HL
0C4C  7C           LD    A,H         ;Vorzeichen negativ?
0C4D  B7           OR    A
0C4E  F2 9A 0A     JP    P,0A9AH     ;nein, HL in X, fertig
0C51  AF           XOR   A           ;A = 0
0C52  4F           LD    C,A         ;C = 0
0C53  95           SUB   L           ;0 - L in L
0C54  6F           LD    L,A
0C55  79           LD    A,C         ;A = 0
0C56  9C           SBC   A,H         ;0 - H in H
0C57  67           LD    H,A
0C58  C3 9A 0A     JP    0A9AH       ;HL in X übertragen

                   ****************************************

                   Negativen Wert einer Integer-Zahl bilden
                   Eing.: X = Argument
                   Ausg.: X = Funktionswert
0C5B  2A 21 79     LD    HL,(7921H)  ;Argument in HL übertragen
0C5E  CD 51 0C     CALL  0C51H       ;0 - Argument in HL und X
0C61  7C           LD    A,H         ;HL = 32768 ?
0C62  EE 80        XOR   80H
```

```
0C64  B5          OR    L
0C65  C0          RET   NZ              ;nein, fertig!
0C66  EB          EX    DE,HL           ;ja, HL in einf.Genauigkeit umwand.
0C67  CD EF 0A    CALL  0AEFH           ;Typ = einf. Genauigkeit setzen
0C6A  AF          XOR   A
0C6B  06 98       LD    B,98H           ;Exponent = 18 setzen
0C6D  C3 69 09    JP    0969H           ;weiter bei 0969H

                  ************************************

                  Subtraktion mit doppelter Genauigkeit
                  Eing.: X = Minuend
                         Y = Subtrahend
                  Ausg.: X = Differenz
0C70  21 2D 79    LD    HL,792DH        ;Adresse MSB Y laden
0C73  7E          LD    A,(HL)          ;Vorzeichen Y invertieren
0C74  EE 80       XOR   80H
0C76  77          LD    (HL),A

                  ************************************

                  Addition mit doppelter Genauigkeit
                  Eing.: X = Summand
                         Y = Summand
                  Ausg.: X = Summe
0C77  21 2E 79    LD    HL,792EH        ;Adresse Exponent Y laden
0C7A  7E          LD    A,(HL)          ;Y = 0 ?
0C7B  B7          OR    A
0C7C  C8          RET   Z               ;ja, X = Ergebnis
0C7D  47          LD    B,A             ;Exponent Y in B
0C7E  2B          DEC   HL              ;Adresse MSB Y
0C7F  4E          LD    C,(HL)          ;Vorzeichen Y in C
0C80  11 24 79    LD    DE,7924H        ;Adresse Exponent X laden
0C83  1A          LD    A,(DE)
0C84  B7          OR    A               ;X = 0 ?
0C85  CA F4 09    JP    Z,09F4H         ;ja, Y nach X als Ergebnis
0C88  90          SUB   B               ;Exponent X >= Exponent Y ?
0C89  30 16       JR    NC,0CA1H        ;ja
0C8B  2F          CPL                   ;nein, Exponentendiff. invertieren
0C8C  3C          INC   A
0C8D  F5          PUSH  AF              ;und auf Stack sichern
                  X und Y vertauschen
0C8E  0E 08       LD    C,8             ;Bytezähler laden
090   23          INC   HL              ;Adresse Exponent Y laden
```

```
0C91  E5           PUSH  HL            ;und auf Stack
0C92  1A           LD    A,(DE)        ;1 Byte vertauschen
0C93  46           LD    B,(HL)
0C94  77           LD    (HL),A
0C95  78           LD    A,B
0C96  12           LD    (DE),A
0C97  1B           DEC   DE            ;Adress-Zeiger - 1
0C98  2B           DEC   HL
0C99  0D           DEC   C             ;fertig ?
0C9A  20 F6        JR    NZ,0C92H      ;nein, nächstes Byte
0C9C  E1           POP   HL            ;Adresse Exponent Y wieder laden
0C9D  46           LD    B,(HL)        ;Exponent Y in B
0C9E  2B           DEC   HL            ;Adresse MSB Y in HL
0C9F  4E           LD    C,(HL)        ;MSB Y in C übertragen
0CA0  F1           POP   AF            ;Exponentendifferenz laden
0CA1  FE 39        CP    39H           ;>= Mantissenlänge + 1 ?
0CA3  D0           RET   NC            ;ja, fertig!
0CA4  F5           PUSH  AF            ;Exponentendifferenz auf Stack
0CA5  CD DF 09     CALL  09DFH         ;Vorzeichenbits entfernen,
                                       ;Vorz.flag des Ergebnisses bilden
0CA8  23           INC   HL            ;Zus. Byte für Rechtsschieben
0CA9  36 00        LD    (HL),0        ;(7926H) löschen
0CAB  47           LD    B,A           ;Vorzeichen-Flag in B
0CAC  F1           POP   AF            ;Exponentendifferenz laden
                                       ;= Verschiebezähler
0CAD  21 2D 79     LD    HL,792DH      ;MSB Y adressieren
0CB0  CD 69 0D     CALL  0D69H         ;Y rechtsschieben
0CB3  3A 26 79     LD    A,(7926H)     ;herausgeschobenes Byte
0CB6  32 1C 79     LD    (791CH),A     ;nach X übertragen
0CB9  78           LD    A,B           ;beide Vorzeichen gleich ?
0CBA  B7           OR    A
0CBB  F2 CF 0C     JP    P,0CCFH       ;nein, Subtraktion

                   Addition der Mantissen
0CBE  CD 33 0D     CALL  0D33H         ;Mantissenaddition, Überlauf ?
0CC1  D2 0E 0D     JP    NC,0D0EH      ;nein, zum Ende
0CC4  EB           EX    DE,HL         ;HL = Adresse Exponent X
0CC5  34           INC   (HL)          ;Exponent X + 1, Überlauf ?
0CC6  CA B2 07     JP    Z,07B2H       ;ja, OVERFLOW - Error
0CC9  CD 90 0D     CALL  0D90H         ;Mantisse 1 Bit rechts schieben
0CCC  C3 0E 0D     JP    0D0EH         ;weiter bei 0D0EH

                   Subtraktion der Mantissen
0CCF  CD 45 0D     CALL  0D45H         ;Mantissensubtraktion
```

```
0CD2  21 25 79        LD     HL,7925H       ;Adresse Vorzeichenflag
0CD5  DC 57 0D        CALL   C,0D57H        ;Unterlauf ? ja, Mantisse X
                                            ;komplementieren

                      ****************************************

                      Normalisieren
0CD8  AF              XOR    A              ;Verschiebezähler = 0
0CD9  47              LD     B,A
0CDA  3A 23 79        LD     A,(7923H)      ;MSB X laden
0CDD  B7              OR     A              ;= 0 ?
0CDE  20 1E           JR     NZ,0CFEH       ;nein !
0CE0  21 1C 79        LD     HL,791CH       ;ja, X um 1 Byte links schieben
0CE3  0E 08           LD     C,8            ;Bytezähler
0CE5  56              LD     D,(HL)         ;Byte laden
0CE6  77              LD     (HL),A         ;letztes Byte an diese Stelle
0CE7  7A              LD     A,D
0CE8  23              INC    HL             ;Adresse erhöhen
0CE9  0D              DEC    C              ;fertig ?
0CEA  20 F9           JR     NZ,0CE5H       ;nein, weiter
0CEC  78              LD     A,B            ;Verschiebezähler - 8
0CED  D6 08           SUB    8
0CEF  FE C0           CP     0C0H           ;40 Verschiebungen? (X = 0)
0CF1  20 E6           JR     NZ,0CD9H       ;nein, weiter
0CF3  C3 78 07        JP     0778H          ;ja, X = 0, fertig!

0CF6  05              DEC    B              ;Verschiebungen - 1
0CF7  21 1C 79        LD     HL,791CH       ;Adresse LSB X laden
0CFA  CD 97 0D        CALL   0D97H          ;X um ein Bit nach links
0CFD  B7              OR     A              ;höchstwertigstes Bit gesetzt?
0CFE  F2 F6 0C        JP     P,0CF6H        ;nein, weiter
0D01  78              LD     A,B            ;Anzahl Verschiebungen = 0 ?
0D02  B7              OR     A
0D03  28 09           JR     Z,0D0EH        ;ja, zum Ende
0D05  21 24 79        LD     HL,7924H       ;Adresse Exponent X
0D08  86              ADD    A,(HL)         ;neuer Exponent = alter Exponent
                                            ;+ Anzahl Verschiebungen
0D09  77              LD     (HL),A         ;zurück in X
0D0A  D2 78 07        JP     NC,0778H       ;Unterlauf? ja, X=0, fertig
0D0D  C8              RET    Z              ;X = 0 ? ja, fertig!

0D0E  3A 1C 79        LD     A,(791CH)      ;höchstw. Bit vom LSB X = 0 ?
0D11  B7              OR     A
0D12  FC 20 0D        CALL   M,0D20H        ;nein, X runden
```

```
0D15  21 25 79        LD    HL,7925H      ;Vorzeichen-Flag adressieren
0D18  7E              LD    A,(HL)        ;laden und Vorzeichen
0D19  E6 80           AND   80H           ;ausblenden
0D1B  2B              DEC   HL            ;Adresse MSB X laden
0D1C  2B              DEC   HL
0D1D  AE              XOR   (HL)          ;Vorzeichen invertieren und mit
                                          ;MSB X verknüpfen
0D1E  77              LD    (HL),A        ;MSB zurück in X
0D1F  C9              RET

                      ************************************

                      Runden
0D20  21 1D 79        LD    HL,791DH      ;Adresse LSB X laden
0D23  06 07           LD    B,7           ;Mantissenlänge = 7 Bytes
0D25  34              INC   (HL)          ;Byteinhalt + 1, überlauf?
0D26  C0              RET   NZ            ;nein, fertig
0D27  23              INC   HL            ;ja, nächstes Byte
0D28  05              DEC   B             ;alle Mantissenbytes?
0D29  20 FA           JR    NZ,0D25H      ;nein, weiter
0D2B  34              INC   (HL)          ;Carry durch ganze Mantisse?
0D2C  CA B2 07        JP    Z,07B2H       ;ja, OVERFLOW - Error
0D2F  2B              DEC   HL            ;MSB X = 80H setzen
0D30  36 80           LD    (HL),80H
0D32  C9              RET

                      ************************************

                      Mantissen-Addition doppelter Genauigkeit
                      Mantisse X = Mantisse X + Mantisse Y
0D33  21 27 79        LD    HL,7927H      ;Adresse LSB Y
0D36  11 1D 79        LD    DE,791DH      ;Adresse LSB X
0D39  0E 07           LD    C,7           ;Zähler = 7 Bytes
0D3B  AF              XOR   A             ;Carry löschen
0D3C  1A              LD    A,(DE)        ;Byte aus X laden
0D3D  8E              ADC   A,(HL)        ;Byte aus Y addieren
0D3E  12              LD    (DE),A        ;Summe in X speichern
0D3F  13              INC   DE            ;Adresszeiger erhöhen
0D40  23              INC   HL
0D41  0D              DEC   C             ;fertig ?
0D42  20 F8           JR    NZ,0D3CH      ;nein, weiter
0D44  C9              RET

                      ************************************
```

```
                        Mantissen - Subtraktion doppelter Genauigkeit
                        Mantisse X = Mantisse X - Mantisse Y
0D45  21 27 79          LD      HL,7927H        ;Adresse LSB Y
0D48  11 1D 79          LD      DE,791DH        ;Adresse LSB X
0D4B  0E 07             LD      C,7             ;7 Bytes als Zähler
0D4D  AF                XOR     A               ;Carry löschen
0D4E  1A                LD      A,(DE)          ;Byte aus X laden
0D4F  9E                SBC     A,(HL)          ;Byte aus Y subtrahieren
0D50  12                LD      (DE),A          ;Differenz in X speichern
0D51  13                INC     DE              ;Adresszeiger + 1
0D52  23                INC     HL
0D53  0D                DEC     C               ;fertig ?
0D54  20 F8             JR      NZ,0D4EH        ;nein, weiter
0D56  C9                RET

                        ****************************************

                        Mantisse von X komplementieren
0D57  7E                LD      A,(HL)          ;Vorzeichen-Flag komplementieren
0D58  2F                CPL
0D59  77                LD      (HL),A
0D5A  21 1C 79          LD      HL,791CH        ;Adresse LSB X laden
0D5D  06 08             LD      B,8             ;Bytezähler = 8
0D5F  AF                XOR     A               ;Carry löschen
0D60  4F                LD      C,A             ;C = 0
0D61  79                LD      A,C             ;A = 0
0D62  9E                SBC     A,(HL)          ;Byte von 0 subtrahieren
0D63  77                LD      (HL),A          ;und zurückspeichern
0D64  23                INC     HL              ;Adresszeiger + 1
0D65  05                DEC     B               ;fertig ?
0D66  20 F9             JR      NZ,0D61H        ;nein, weiter
0D68  C9                RET

                        ****************************************

                        8 Bytes rechts schieben
                        Eing.: A = Anzahl der zu versch. Bits
                               HL = Adresse MSB d. zu versch. Bereichs
                               C = Inhalt MSB
0D69  71                LD      (HL),C          ;MSB abspeichern
0D6A  E5                PUSH    HL              ;MSB-Adresse auf Stack
0D6B  D6 08             SUB     8               ;mehr als 8 Verschiebungen?
0D6D  38 0E             JR      C,0D7DH         ;nein !
```

```
0D6F  E1          POP   HL          ;Adresse aus Stack zurück
0D70  E5          PUSH  HL          ;und wieder auf Stack
0D71  11 00 08    LD    DE,0800H    ;Bytezähler = 8 (D)
                                    ;Zwischenspeicher löschen (E)
0D74  4E          LD    C,(HL)      ;Byte nach C laden
0D75  73          LD    (HL),E      ;letztes Byte aus Zwischenspeicher
                                    ;eintragen
0D76  59          LD    E,C         ;Byte aus C in Zwischenspeicher
0D77  2B          DEC   HL          ;Adresszeiger - 1
0D78  15          DEC   D           ;Bytezähler - 1
0D79  20 F9       JR    NZ,0D74H    ;fertig ? nein-zurück
0D7B  18 EE       JR    0D6BH       ;nächste Byteverschiebung

0D7D  C6 09       ADD   A,9         ;Bitverschiebungen + 1
0D7F  57          LD    D,A         ;in D
0D80  AF          XOR   A           ;Carry löschen
0D81  E1          POP   HL          ;Adresszeiger aus Stack laden
0D82  15          DEC   D           ;noch eine Verschiebung ?
0D83  C8          RET   Z           ;nein, fertig
0D84  E5          PUSH  HL          ;Adresszeiger retten
0D85  1E 08       LD    E,8         ;Bytezähler = 8
0D87  7E          LD    A,(HL)      ;Byte laden
0D88  1F          RRA               ;1 Bit rechts schieben
0D89  77          LD    (HL),A      ;und zurück in Speicher
0D8A  2B          DEC   HL          ;Adresszeiger -1
0D8B  1D          DEC   E           ;Bytezähler -1
0D8C  20 F9       JR    NZ,0D87H    ;fertig ? nein-zurück
0D8E  18 F0       JR    0D80H       ;nächste Bitverschiebung

                  ************************************

                  X - Register um 1 Bit rechts schieben
0D90  21 23 79    LD    HL,7923H    ;MSB X adressieren
0D93  16 01       LD    D,1         ;Bitzähler = 1
0D95  18 ED       JR    0D84H       ;weiter bei 0D84H

                  ************************************

                  Speicherbereich 1 Bit links schieben
                  Eing.: HL = Anfangsadresse des Bereichs
                         Carry = nachzuschiebendes Bit
0D97  0E 08       LD    C,8         ;Bytezähler = 8
0D99  7E          LD    A,(HL)      ;Byte laden
0D9A  17          RLA               ;nach links schieben
```

```
0D9B  77          LD    (HL),A      ;und zurück in Speicher
0D9C  23          INC   HL          ;Adresszeiger + 1
0D9D  0D          DEC   C           ;Bytezähler - 1
0D9E  20 F9       JR    NZ,0D99H    ;fertig ? nein-zurück
0DA0  C9          RET

                  ****************************************

                  Multiplikation doppelter Genauigkeit
                  Eing.: X = Faktor
                         Y = Faktor
                  Ausg.: X = Produkt
0DA1  CD 55 09    CALL  0955H       ;1. Faktor = 0 ?
0DA4  C8          RET   Z           ;ja, fertig
0DA5  CD 0A 09    CALL  090AH       ;Exponent und Vorzeichen verarb.
0DA8  CD 39 0E    CALL  0E39H       ;Mantisse 1.Faktor aus X nach 414A-
                                    ;4150 übertragen. X löschen.
0DAB  71          LD    (HL),C      ;LSB X löschen
0DAC  13          INC   DE          ;Adresse LSB 1. Faktor
0DAD  06 07       LD    B,7         ;Bytezähler = 7
0DAF  1A          LD    A,(DE)      ;Byte von 1. Faktor laden
0DB0  13          INC   DE          ;Adresszeiger 1. Faktor + 1
0DB1  B7          OR    A           ;= 0 ?
0DB2  D5          PUSH  DE          ;Adresszeiger auf Stack
0DB3  28 17       JR    Z,0DCCH     ;Byte ist 0!
0DB5  0E 08       LD    C,8         ;nicht 0, Bitzähler = 8
0DB7  C5          PUSH  BC          ;Bitzähler retten
0DB8  1F          RRA               ;nächstes Bit gesetzt?
0DB9  47          LD    B,A         ;Byte in B übertragen
0DBA  DC 33 0D    CALL  C,0D33H     ;ja, 2. Faktor auf X addieren
0DBD  CD 90 0D    CALL  0D90H       ;X ein Bit rechts rotieren
0DC0  78          LD    A,B         ;Byte von B wieder in A zurück
0DC1  C1          POP   BC          ;Bytezähler wieder laden
0DC2  0D          DEC   C           ;Byte fertig bearbeitet?
0DC3  20 F2       JR    NZ,0DB7H    ;nein, nächstes Bit
0DC5  D1          POP   DE          ;Adresszeiger wieder laden
0DC6  05          DEC   B           ;alle 7 Bytes bearbeitet?
0DC7  20 E6       JR    NZ,0DAFH    ;nein, nächstes Byte
0DC9  C3 D8 0C    JP    0CD8H       ;zur Normalisierung
0DCC  21 23 79    LD    HL,7923H    ;Ergebnis um 1 Byte rechtsschieben
0DCF  CD 70 0D    CALL  0D70
0DD2  18 F1       JR    0DC5H       ;nächstes Byte

                  ****************************************
```

```
0DD4  00 00 00 00                        Konstante 10 (dopp. Genauigkeit)
0DD8  00 00 20 84                        Konstante 10 (einf. Genauigkeit)

                  ***************************************

                  Division durch 10 mit doppelter Genauigkeit
                  Eing.: X = Divident
                  Ausg.: X = Quotient
0DDC  11 D4 0D    LD      DE,0DD4H       ;Adresse der Konstante 10 laden
0DDF  21 27 79    LD      HL,7927H       ;Adresse von Y laden
0DE2  CD D3 09    CALL    09D3H          ;Konstante 10 in Y

                  ***************************************

                  Division mit doppelter Genauigkeit
                  Eing.: X = Divident
                         Y = Divisor
                  Ausg.: X = Quotient
0DE5  3A 2E 79    LD      A,(792EH)      ;Divisor = 0 ?
0DE8  B7          OR      A
0DE9  CA 9A 19    JP      Z,199AH        ;ja, DIVISION BY ZERO - Error
0DEC  CD 07 09    CALL    0907H          ;Vorzeichen und Exponent verarb.
0DEF  34          INC     HL             ;Exponent-Korrektur (+2)
0DF0  34          INC     HL             ;(0907 erg. Exp X - Exp Y - 1)
0DF1  CD 39 0E    CALL    0E39H          ;Divident in Bereich 414A-4150.
                                         ;X für Ergebnis löschen
0DF4  21 51 79    LD      HL,7951H       ;höchstw. Byte des Dividenten = 0
0DF7  71          LD      (HL),C
0DF8  41          LD      B,C            ;Flag löschen
0DF9  11 4A 79    LD      DE,794AH       ;Divident adressieren
0DFC  21 27 79    LD      HL,7927H       ;Divisor adressieren
0DFF  CD 4B 0D    CALL    0D4BH          ;Divident-Divisor in Divident
0E02  1A          LD      A,(DE)         ;MSB Divident laden
0E03  99          SBC     A,C            ;- Carry (C=0)
0E04  3F          CCF                    ;Carry invertieren, Unterlauf?
0E05  38 0B       JR      C,0E12H        ;nein, 1 in Ergebnis schieben
0E07  11 4A 79    LD      DE,794AH       ;ja, Subtraktion rückgängig machen
0E0A  21 27 79    LD      HL,7927H       ;DE=Divident, HL=Divisor
0E0D  CD 39 0D    CALL    0D39H          ;Divident + Divisor in Divident
0E10  AF          XOR     A              ;Carry löschen
0E11  DA          DEFB    0DAH           ;JP C,0412H  Dummy, wird nie ausgef
0E12  12          LD      (DE),A         ;MSB Divident speichern
0E13  04          INC     B              ;Flag setzen
```

```
0E14  3A 23 79     LD    A,(7923H)      ;MSB Ergebnis laden
0E17  3C           INC   A              ;Bit 7 gesetzt ?
0E18  3D           DEC   A
0E19  1F           RRA                  ;ermitteltes Bit in A(7) zum runden
0E1A  FA 11 0D     JP    M,0D11H        ;fertig, zur Rundung
0E1D  17           RLA                  ;Bit wieder in Carry schieben
0E1E  21 1D 79     LD    HL,791DH       ;LSB Ergebnis (X) adressieren
0E21  0E 07        LD    C,7            ;Bytezähler = 7
0E23  CD 99 0D     CALL  0D99H          ;Ergebnis links rotieren,
                                        ;Bit einschieben.
0E26  21 4A 79     LD    HL,794AH       ;Adresse Divident
0E29  CD 97 0D     CALL  0D97H          ;Divident 1 Bit links rotieren
0E2C  78           LD    A,B            ;Flag gesetzt ?
0E2D  B7           OR    A
0E2E  20 C9        JR    NZ,0DF9H       ;ja, weiter
0E30  21 24 79     LD    HL,7924H       ;nein, Exponent Ergebnis - 1
0E33  35           DEC   (HL)           ;Unterlauf ?
0E34  20 C3        JR    NZ,0DF9H       ;nein, weiter
0E36  C3 B2 07     JP    07B2H          ;ja, OVERFLOW - Error

                   ************************************

                   Unterprogramm für Multiplikation und Division
                   doppelter Genauigkeit
0E39  79           LD    A,C            ;MSB Y in Speicher
0E3A  32 2D 79     LD    (792DH),A
0E3D  2B           DEC   HL             ;Adresse MSB X laden
0E3E  11 50 79     LD    DE,7950H       ;Zeiger auf Hilfsregister
0E41  01 00 07     LD    BC,0700H       ;X in Hilfsregister übertragen
                                        ;X löschen, Bytezähler = 7
0E44  7E           LD    A,(HL)         ;Byte aus X laden
0E45  12           LD    (DE),A         ;in Hilfsregister
0E46  71           LD    (HL),C         ;Byte in X löschen
0E47  1B           DEC   DE             ;Adresszeiger -1
0E48  2B           DEC   HL
0E49  05           DEC   B              ;Bytezähler - 1
0E4A  20 F8        JR    NZ,0E44H       ;fertig ? nein-zurück
0E4C  C9           RET

                   ************************************

                   Multiplikation doppelter Genauigkeit mit 10
                   Eing.: X = Faktor
                   Ausg.: X = Produkt
```

```
0E4D  CD FC 09        CALL    09FCH           ;X nach Y übertragen
0E50  EB              EX      DE,HL           ;Adresse Exponent X laden
0E51  2B              DEC     HL
0E52  7E              LD      A,(HL)          ;Faktor in X = 0 ?
0E53  B7              OR      A
0E54  C8              RET     Z               ;ja, Ergebnis = 0
0E55  C6 02           ADD     A,2             ;Exponent X + 2 (Faktor * 4)
0E57  DA B2 07        JP      C,07B2H         ;überlauf ? ja, OVERFLOW - Error
0E5A  77              LD      (HL),A          ;Exponent X zurückspeichern
0E5B  E5              PUSH    HL              ;Adresse Exponent X auf Stack
0E5C  CD 77 0C        CALL    0C77H           ;X + Y nach X (X = Faktor * 5)
0E5F  E1              POP     HL              ;Adresse Exponent X vom Stack
0E60  34              INC     (HL)            ;Exponent X + 1 (X = Faktor * 10)
0E61  C0              RET     NZ              ;überlauf? nein-fertig
0E62  C3 B2 07        JP      07B2H           ;ja, OVERFLOW - Error

                      ****************************************

                      String in Zahl doppelter Genauigkeit umwandeln
                      Eing.: HL = Adresse des Strings
                      Ausg.: X = Zahl
0E65  CD 78 07        CALL    0778H           ;X = 0
0E68  CD EC 0A        CALL    0AECH           ;Typ = doppelte Genauigkeit
0E6B  F6 AF           OR      0AFH            ;Zero-Flag = 0

                      ****************************************

                      String in Zahl passenden Typs umwandeln
                      Eing.: HL = Adresse des Strings
                      Ausg.: X = Zahl
0E6C  AF              XOR     A               ;Zero-Flag = 1
                                              ;(Byte auch im Befehl E6B)
0E6D  EB              EX      DE,HL           ;Adresszeiger in DE
0E6E  01 FF 00        LD      BC,00FFH        ;Nachkommastellen = 0
                                              ;'.'-Flag = FF
0E71  60              LD      H,B             ;HL = 0
0E72  68              LD      L,B
0E73  CC 9A 0A        CALL    Z,0A9AH         ;Ansprung bei 0E6C? ja, Typ=Integer
0E76  EB              EX      DE,HL           ;Adresszeiger wieder in HL
0E77  7E              LD      A,(HL)          ;Zeichen laden
0E78  FE 2D           CP      2DH             ;= '-' ? ja - Z-Flag = 1
0E7A  F5              PUSH    AF              ;Flag auf Stack
0E7B  CA 83 0E        JP      Z,0E83H         ;ja, nächstes Zeichen
0E7E  FE 2B           CP      2BH             ;= '+' ?
```

```
0E80  28 01        JR    Z,0E83H      ;ja, nächstes Zeichen
0E82  2B           DEC   HL          ;kein Vorzeichen, Zeiger - 1
0E83  D7           RST   10H         ;folgt eine Ziffer?
0E84  DA 29 0F     JP    C,0F29H      ;ja!
0E87  FE 2E        CP    2EH          ;= '.' ?
0E89  CA E4 0E     JP    Z,0EE4H      ;ja!
0E8C  FE 45        CP    45H          ;= 'E'? (Exponent bei einf. Gen.)
0E8E  28 14        JR    Z,0EA4       ;ja!
0E90  FE 25        CP    25H          ;= '%'? (Zahl als Integer betracht)
0E92  CA EE 0E     JP    Z,0EEEH      ;ja!
0E95  FE 23        CP    23H          ;= '#'? (Zahl als dopp. Gen. betr.)
0E97  CA F5 0E     JP    0EF5H        ;ja!
0E9A  FE 21        CP    21H          ;= '!'? (Zahl als einf. Gen. betr.)
0E9C  CA F6 0E     JP    Z,0EF6H      ;ja!
0E9F  FE 44        CP    44H          ;= 'D'? (Exponent bei dopp. Gen.)
0EA1  20 24        JR    NZ,0EC7H     ;nein!

                   Exponent ermitteln
0EA3  B7           OR    A            ;Flag für Typanpassung setzen
0EA4  CD FB 0E     CALL  0EFBH        ;Zahl in einf. o. dopp. Genauigkeit
0EA7  E5           PUSH  HL           ;Adresszeiger retten
0EA8  21 BD 0E     LD    HL,0EBDH     ;Rücksprungadresse laden
0EAB  E3           EX    (SP),HL      ;im Stack mit Adr.zeiger tauschen
0EAC  D7           RST   10H          ;nächstes Zeichen
0EAD  15           DEC   D            ;Exp.-Vorz.-Flag auf '-'
0EAE  FE CE        CP    0CEH         ;= '-'? (Token)
0EB0  C8           RET   Z            ;ja!
0EB1  FE 2D        CP    2DH          ;= '-'?
0EB3  C8           RET   Z            ;ja!
0EB4  14           INC   D            ;Exp.-Vorz.-Flag auf '+'
0EB5  FE CD        CP    0CDH         ;= '+'? (Token)
0EB7  C8           RET   Z            ;ja!
0EB8  FE 2B        CP    2BH          ;= '+'?
0EBA  C8           RET   Z            ;ja!
0EBB  2B           DEC   HL           ;kein Vorzeichen, Adr.zeiger zur.
0EBC  F1           POP   AF           ;Rücksprungadresse vom Stack entf.
0EBD  D7           RST   10H          ;nächstes Zeichen laden
0EBE  DA 94 0F     JP    C,0F94H      ;Ziffer ? ja-weiter bei 0F94H
0EC1  14           INC   D            ;nein, Exp.-Vorz.-Flag = '-' ?
0EC2  20 03        JR    NZ,0EC7H     ;nein!
0EC4  AF           XOR   A            ;ja, Exponent invertieren
0EC5  93           SUB   E
0EC6  5F           LD    E,A          ;und zurück in E
0EC7  E5           PUSH  HL           ;Adresszeiger retten
```

```
0EC8  7B          LD    A,E          ;Exponent-Nachkommastellen
0EC9  90          SUB   B            ;Differenz > 0 ?
0ECA  FA 0A 0F    CALL  P,0F0AH      ;ja, Zahl*10, Differenz - 1
0ECD  FC 18 0F    CALL  M,0F18H      ;nein, Zahl/10, Differenz + 1
0ED0  20 F8       JR    NZ,0ECAH     ;wiederholen, bis Differenz = 0
0ED2  E1          POP   HL           ;Adresszeiger laden
0ED3  F1          POP   AF           ;Vorzeichen-Flag laden
0ED4  E5          PUSH  HL           ;Adresszeiger wieder auf Stack
0ED5  CC 7B 09    CALL  Z,097BH      ;Vorzeichen-Flag='-'? ja, X = -X
0ED8  E1          POP   HL           ;Adresszeiger zurück
0ED9  E7          RST   20H          ;Typ testen
0EDA  E8          RET   PE           ;doppelte Genauigk.? ja-fertig
0EDB  E5          PUSH  HL           ;Adresszeiger auf Stack
0EDC  21 90 08    LD    HL,0890H     ;Rücksprungadresse auf Stack
0EDF  E5          PUSH  HL
0EE0  CD A3 0A    CALL  0AA3H        ;einf. Genauigkeit, wenn möglich,
                                     ;in Integer umwandeln.
0EE3  C9          RET                ;weiter bei 0890H

                  Dezimalpunkt verarbeiten
0EE4  E7          RST   20H          ;Typ testen
0EE5  0C          INC   C            ;'.'-Flag = 0? (war schon ein '.'?)
0EE6  20 DF       JR    NZ,0EC7H     ;ja, fertig
0EE8  DC FB 0E    CALL  C,0EFBH      ;einf.Genauigkeit!
                                     ;Integer in einf. Genauigkeit umw.
0EEB  C3 83 0E    JP    0E83H        ;nächstes Zeichen

                  '%' gefunden
0EEE  E7          RST   20H          ;Typ testen
0EEF  F2 97 19    JP    P,1997H      ;kein Integer, SYNTAX - Error
0EF2  23          INC   HL           ;Adresszeiger + 1
0EF3  18 D2       JR    0EC7H        ;fertig!

                  '#' gefunden
0EF5  B7          OR    A            ;Flag für Typanpassung setzen

                  '!' gefunden
0EF6  CD FB 0E    CALL  0EFBH        ;X in einf. o. dopp. Genauigk. umw.
0EF9  18 F7       JR    0EF2H        ;weiter bei 0EF2H

                  ##########################################

                  Zahl in einfache oder doppelte Genauigkeit umwandeln
                  Eing.: X = Ausgangszahl
```

```
                        Z-Flag = 1 Umwandlung in einfache Genauigkeit
                        Z-Flag = 0 Umwandlung in doppelte Genauigkeit
                    Ausg.: X = Ergebnis
0EFB  E5            PUSH    HL              ;Register auf Stack retten
0EFC  D5            PUSH    DE
0EFD  C5            PUSH    BC
0EFE  F5            PUSH    AF              ;Flag retten
0EFF  CC B1 0A      CALL    Z,0AB1H         ;Z-Flag=1, in einf. Gen. umwandeln
0F02  F1            POP     AF              ;Flag wieder laden
0F03  C4 DB 0A      CALL    NZ,0ADBH        ;Z-Flag=0, in dopp. Gen. umwandeln
0F06  C1            POP     BC              ;Register wiederherstellen
0F07  D1            POP     DE
0F08  E1            POP     HL
0F09  C9            RET

                    ****************************************

                    Reele Zahl mit 10 multiplizieren
                    Eing.: X = Ausgangswert
                        Z-Flag = 0
                    Ausg.: X = Produkt
0F0A  C8            RET     Z               ;Z-Flag = 1?, zurück
0F0B  F5            PUSH    AF              ;A auf Stack
0F0C  E7            RST     20H             ;Typ testen
0F0D  F5            PUSH    AF              ;Typ-Flag merken
0F0E  E4 3E 09      CALL    PO,093EH        ;dopp.Gen.? => * 10
0F11  F1            PUSH    AF              ;Typ-Flag wieder laden
0F12  EC 4D 0E      CALL    PE,0E4DH        ;einf.Gen.? => * 10
0F15  F1            POP     AF              ;A-Reg wiederherstellen
0F16  3D            DEC     A               ;A - 1
0F17  C9            RET

                    ****************************************

                    Reele Zahl durch 10 dividieren
                    Eing.: X = Ausgangswert
                    Ausg.: X = Quotient
0F18  D5            PUSH    DE              ;Register retten
0F19  E5            PUSH    HL
0F1A  F5            PUSH    AF
0F1B  E7            RST     20H             ;Typ testen
0F1C  F5            PUSH    AF              ;Typ-Flag sichern
0F1D  E4 97 08      CALL    PO,0897H        ;Typ=einf.Gen.? => /10
0F20  F1            POP     AF              ;Typ-Flag laden
```

```
0F21  EC DC 0D      CALL  0DDCH       ;Typ=dopp.Gen.? => /10
0F24  F1            POP   AF          ;Registerinhalte wiederherstellen
0F25  E1            POP   HL
0F26  D1            POP   DE
0F27  3C            INC   A           ;A + 1
0F28  C9            RET

                    ************************************

                    Ziffer verarbeiten
0F29  D5            PUSH  DE          ;Exp.-Vorz.-Flag u. Exponent retten
0F2A  78            LD    A,B         ;Nachkommast. + 1, wenn '.'-Flag 0
0F2B  89            ADC   A,C         ;(Carry ist gesetzt)
0F2C  47            LD    B,A         ;in B
0F2D  C5            PUSH  BC          ;Nachkommast. und Flag retten
0F2E  E5            PUSH  HL          ;Adresszeiger retten
0F2F  7E            LD    A,(HL)      ;Ziffer laden
0F30  D6 30         SUB   30H         ;Zonenteil entfernen
0F32  F5            PUSH  AF          ;bereinigte Ziffer auf Stack
0F33  E7            RST   20H         ;Typ testen
0F34  F2 5D 0F      JP    P,0F5DH     ;einf. o. dopp. Genauigkeit !

                    Integer
0F37  2A 21 79      LD    HL,(7921H)  ;Wert aus X laden
0F3A  11 CD 0C      LD    DE,0CCDH    ;>= 3277 ? (d.h. 10 * X >=32770)
0F3D  DF            RST   18H         ;DE mit HL vergleichen
0F3E  30 19         JR    NC,0F59H    ;ja, in einf. Genauigkeit umwand.
0F40  54            LD    D,H         ;Zahl mit 10 multiplizieren
0F41  5D            LD    E,L
0F42  29            ADD   HL,HL       ;* 2
0F43  29            ADD   HL,HL       ;* 4
0F44  19            ADD   HL,DE       ;* 5
0F45  29            ADD   HL,DE       ;* 10
0F46  F1            POP   AF          ;Ziffer wieder laden
0F47  4F            LD    C,A         ;in BC (B = 0)
0F48  09            ADD   HL,BC       ;und auf Zahl addieren
0F49  7C            LD    A,H         ;neue Zahl > 32767 ?
0F4A  B7            OR    A
0F4B  FA 57 0F      JP    M,0F57H     ;ja, in einf. Genauigkeit umw.
0F4E  22 21 79      LD    (7921H),HL  ;neue Zahl zurück in X
0F51  E1            POP   HL          ;Adresszeiger wieder laden
0F52  C1            POP   BC          ;Nachkommast. + Flag zurück
0F53  D1            POP   DE          ;Exp.-Vorz.-Flag + Exponent zurück
0F54  C3 83 0E      JP    0E83H       ;nächstes Zeichen
```

```
0F57  79           LD    A,C          ;Ziffer retten
0F58  F5           PUSH  AF
0F59  CD CC 0A     CALL  0ACCH        ;HL in einf. Genauigkeit umw.
0F5C  37           SCF                ;n.Sprungbefehl ignorieren

0F5D  30 18        JR    NC,0F77H     ;doppelte Genauigkeit ? ja-Sprung!

                   Zahl einfacher Genauigkeit
0F5F  01 74 94     LD    BC,9474H     ;Konstante 1E6 in Y
0F62  11 00 24     LD    DE,2400H
0F65  CD 0C 0A     CALL  0A0CH        ;Zahl >= 1E6 ?
0F68  F2 74 0F     JP    P,0F74H      ;ja, in dopp. Genauigk. umwandeln
0F6B  CD 3E 09     CALL  093EH        ;Zahl * 10
0F6E  F1           POP   AF           ;Ziffer wieder laden
0F6F  CD 89 0F     CALL  0F89H        ;und zur Zahl addieren
0F72  18 DD        JR    0F51H        ;zurück

                   Zahl doppelter Genauigkeit
0F74  CD E3 0A     CALL  0AE3H        ;Zahl in dopp. Genauigk. umwandeln
0F77  CD 4D 0E     CALL  0E4DH        ;Zahl * 10
0F7A  CD FC 09     CALL  09FCH        ;Zahl nach Y übertragen
0F7D  F1           POP   AF           ;Ziffer laden
0F7E  CD 64 09     CALL  0964H        ;in X übertragen
0F81  CD E3 0A     CALL  0AE3H        ;in dopp. Genauigkeit umwandeln
0F84  CD 77 0C     CALL  0C77H        ;und auf Zahl addieren
0F87  18 C8        JR    0F51H        ;zurück

                   ************************************

                   8 Bit Integer auf Zahl einfacher Genauigkeit addieren
                   Eing.: X = 1. Summand (einf. Genauigk.)
                          A = 2. Summand (8-Bit Integer mit Vorzeichen)
                   Ausg.: X = Summe
0F89  CD A4 09     CALL  09A4H        ;1. Summand auf Stack retten
0F8C  CD 64 09     CALL  0964H        ;2. Summand mit einf. Gen. in X
0F8F  C1           POP   BC           ;1. Summand vom Stack in Y
0F90  D1           POP   DE
0F91  C3 16 07     JP    0716H        ;Summe bilden

                   ************************************

                   Exponenten - Ziffer verarbeiten
0F94  7B           LD    A,E          ;Exponent > 9 ?
```

```
0F95  FE 0A           CP    10
0F97  30 09           JR    NC,0FA2H      ;ja, überlauf erzeugen
0F99  07              RLCA                ;Exponent * 10
0F9A  07              RLCA
0F9B  83              ADD   A,E
0F9C  07              RLCA
0F9D  86              ADD   A,(HL)        ;Zeichen addieren
0F9E  D6 30           SUB   30H           ;Zonenteil eliminieren
0FA0  5F              LD    E,A           ;= neuer Exponent
0FA1  FA              DEFB  0FAH          ;JP M,321EH Dummy, wird nie ausgef.
0FA2  1E 32           LD    E,32H         ;Exponent = 32, ruft überlauf herv.
0FA4  C3 BD 0E        JP    0EBDH         ;nächste Ziffer verarbeiten

                      ************************************

                      Zur Ergänzung einer Fehlermeldung
                      ' IN ' Zeilennummer ausgeben
                      Eing.: HL = Zeilennummer
0FA7  E5              PUSH  HL            ;Zeilennummer auf Stack
0FA8  21 24 19        LD    HL,1924H      ;Textadresse 'IN' laden
0FAB  CD A7 28        CALL  28A7H         ;Text ausgeben
0FAE  E1              POP   HL            ;Zeilennummer wieder laden

                      Zeilennummer ausgeben
                      Eing.: HL = Zeilennummer
0FAF  CD 9A 0A        CALL  0A9AH         ;Zeilennummer als Integer in X
0FB2  AF              XOR   A             ;Format-Flag löschen
0FB3  CD 34 10        CALL  1034H         ;speichern u. Puffer initialisieren
0FB6  B6              OR    (HL)          ;X als Integer ohne Vorzeichen
0FB7  CD D9 0F        CALL  0FD9H         ;in String umwandeln
0FBA  C3 A6 28        JP    28A6H         ;String ausgeben

                      ************************************

                      Zahl in formatierten String umwandeln
                      Eing.: X = Zahl
                             A = Format-Flag
                                 Bit 0 - Exponentenausgabe
                                 Bit 1 - nicht verwendet
                                 Bit 2 - Vorzeichen hinter Zahl
                                 Bit 3 - '+' auch ausgeben
                                 Bit 4 - '$' vor Zahl ausgeben
                                 Bit 5 - führende Leerzeichen durch '*' ersetz.
                                 Bit 6 - ',' zur Tausender-Trennung ausgeben
```

```
                                Bit 7 - Formatierung durchführen
                              B = Anzahl der Vorkommastellen
                              C = Anzahl der Nachkommastellen + 1 (für '.')
                            Ausg.: (HL)... = formatierter String mit X'00' abgeschl.
0FBE  CD 34 10      CALL    1034H           ;Pufferanfang (7930H) adressieren
                                            ;Format-Flag in 78D8H
0FC1  E6 08         AND     8               ;'+' ausgeben ?
0FC3  28 02         JR      Z,0FC7H         ;nein!
0FC5  36 2B         LD      (HL),2BH        ;'+' in Puffer
0FC7  EB            EX      DE,HL           ;Pufferzeiger in DE
0FC8  CD 94 09      CALL    0994H           ;Zahl >= 0?
0FCB  EB            EX      DE,HL           ;Pufferzeiger wieder in HL
0FCC  F2 D9 0F      JP      P,0FD9H         ;ja!
0FCF  36 2D         LD      (HL),2DH        ;'-' in Puffer
0FD1  C5            PUSH    BC              ;Längen-Parameter auf Stack
0FD2  E5            PUSH    HL              ;Pufferzeiger auf Stack
0FD3  CD 7B 09      CALL    097BH           ;Vorzeichen entfernen
0FD6  E1            POP     HL              ;Pufferzeiger laden
0FD7  C1            POP     BC              ;Längen-Parameter laden
0FD8  B4            OR      H               ;Null-Flag zurücksetzen
0FD9  23            INC     HL              ;Pufferzeiger hinter Vorzeichen
0FDA  36 30         LD      (HL),'0'        ;'0' in Puffer
0FDC  3A D8 78      LD      A,(78D8H)       ;Format-Flag in D
0FDF  57            LD      D,A
0FE0  17            RLA                     ;Formatierung durchführen?
0FE1  3A AF 78      LD      A,(78AFH)       ;Typ laden
0FE4  DA 9A 10      JP      C,109AH         ;ja !
0FE7  CA 92 10      JP      Z,1092H         ;Zahl = 0, fertig
0FEA  FE 04         CP      4               ;einf. o. dopp. Genauigkeit?
0FEC  D2 3D 10      JP      NC,103DH        ;ja!

                    Integer in String umwandeln
0FEF  01 00 00      LD      BC,0            ;Parameter für '.' u. ',' löschen
0FF2  CD 2F 13      CALL    132FH           ;String erzeugen

                    Format-Flag Bits 2-5 verarbeiten
0FF5  21 30 79      LD      HL,7930H        ;Pufferzeiger auf Anfang
0FF8  46            LD      B,(HL)          ;Vorzeichen laden
0FF9  0E 20         LD      C,' '           ;Leerzeichen nach Füllzeichen
0FFB  3A D8 78      LD      A,(78D8H)       ;Format-Flag laden
0FFE  5F            LD      E,A             ;in E
0FFF  E6 20         AND     20H             ;mit '*' ausfüllen? (Bit 5)
1001  28 07         JR      Z,100AH         ;nein!
1003  78            LD      A,B             ;Vorzeichen = Leerzeichen?
```

```
1004  B9              CP    C
1005  0E 2A           LD    C,2AH          ;Füllzeichen = '*'
1007  20 01           JR    NZ,100AH       ;nein!
1009  41              LD    B,C            ;Vorzeichen = Füllzeichen
100A  71              LD    (HL),C         ;Füllzeichen in Puffer
100B  D7              RST   10H            ;nächstes Zeichen = Zeilenende ?
100C  28 14           JR    Z,1022H        ;ja, nicht weiter füllen
100E  FE 45           CP    45H            ;=Exp.kennung f. einf.Genauigk.?
1010  28 10           JR    Z,1022H        ;ja, nicht weiter füllen
1012  FE 44           CP    44H            ;=Exp.kennung f. dopp.Genauigk.?
1014  28 0C           JR    Z,1022H        ;ja, nicht weiter füllen
1016  FE 30           CP    30H            ;= '0' ?
1018  28 F0           JR    Z,100AH        ;ja, weiterfüllen
101A  FE 2C           CP    2CH            ;= ',' ?
101C  28 EC           JR    Z,100AH        ;ja, weiterfüllen
101E  FE 2E           CP    2EH            ;= '.' ?
1020  20 03           JR    NZ,1025H       ;nein, nicht weiter füllen
1022  2B              DEC   HL             ;vor '.','E' u. 'D' eine '0' einf.
1023  36 30           LD    (HL),'0'
1025  7B              LD    A,E            ;Dollarzeichen vor Zahl ?
1026  E6 10           AND   10H            ;(Bit 4 des Format-Flags)
1028  28 03           JR    Z,102DH        ;nein!
102A  2B              DEC   HL             ;Pufferzeiger - 1
102B  36 24           LD    (HL),'$'       ;'$' in Puffer
102D  7B              LD    A,E            ;Vorzeichen hinter die Zahl?
102E  E6 04           AND   4              ;(Bit 2 des Format-Flags)
1030  C0              RET   NZ             ;ja, zurück
1031  2B              DEC   HL             ;Pufferzeiger vor Zahl
1032  70              LD    (HL),B         ;Vorzeichen vor die Zahl
1033  C9              RET

                      Puffer initialisieren und Format-Flag speichern
1034  32 D8 78        LD    (78D8H),A      ;Format-Flag speichern
1037  21 30 79        LD    HL,7930H       ;Pufferanfang adressieren
103A  36 20           LD    (HL),' '       ;Leerzeichen an Pufferanfang
103C  C9              RET

                      Zahl einfacher oder doppelter Genauigkeit
                      unformatiert in String umwandeln.
103D  FE 05           CP    5              ;Anzahl Stellen ermitteln
                                           ;einfache Genauigk.? Carry=1
103F  E5              PUSH  HL             ;Pufferzeiger auf Stack
1040  DE 00           SBC   A,0            ;Typ - Carry in A
1042  17              RLA                  ;* 2 = Anzahl Stellen
```

```
                                        ;(einf. = 6, dopp. = 16)
1043  57          LD      D,A           ;Anzahl Stellen in D
1044  14          INC     D             ;+ 1
1045  CD 01 12    CALL    1201H         ;10er Exponent ermitteln.
1048  01 00 03    LD      BC,0300H      ;Parameter für '.' und ',' setzen
104B  82          ADD     A,D           ;Exponent + 2 >= 0 ?
104C  FA 57 10    JP      M,1057H       ;nein, Exponent in Puffer
104F  14          INC     D             ;Anzahl Stellen + 2 in D
1050  BA          CP      D             ;Exponent < Anzahl Stellen ?
1051  30 04       JR      NC,1057H      ;nein, Exponent in Puffer
1053  3C          INC     A             ;ja, Exponent +3 = Dezimalpunkt
1054  47          LD      B,A
1055  3E 02       LD      A,2           ;es wird kein Exponent ausgegeben
1057  D6 02       SUB     2             ;Exponent - 2 in A
1059  E1          POP     HL            ;Pufferzeiger wieder laden
105A  F5          PUSH    AF            ;Exponent auf Stack
105B  CD 91 12    CALL    1291H         ;'.' und ',' setzen
105E  36 30       LD      (HL),'0'      ;'0' in Puffer
1060  CC C9 09    CALL    Z,09C9H       ;'.' gesetzt ? ja, Pufferzeiger + 1
1063  CD A4 12    CALL    12A4H         ;Mantisse in String umwandeln
1066  2B          DEC     HL            ;Pufferzeiger - 1
1067  7E          LD      A,(HL)        ;Zeichen laden
1068  FE 30       CP      '0'           ;= '0' ?
106A  28 FA       JR      Z,1066H       ;ja, weiter
106C  FE 2E       CP      2EH           ;vor der letzten Null '.'?
106E  C4 C9 09    CALL    NZ,09C9H      ;nein! Pufferzeiger + 1
1071  F1          POP     AF            ;Exponent laden. = 0 ?
1072  28 1F       JR      Z,1093H       ;ja, kein Exponent in Puffer
1074  F5          PUSH    AF            ;Exponent wieder auf Stack
1075  E7          RST     20H           ;Typ testen, bei einf. Gen. Carry=1
1076  3E 22       LD      A,22H         ;'D' / 2 laden
1078  8F          ADC     A,A           ;Exp.kennung = 'D' oder 'E'
1079  77          LD      (HL),A        ;in Puffer eintragen
107A  23          INC     HL            ;Pufferzeiger + 1
107B  F1          POP     AF            ;Exponent laden. < 0 ?
107C  36 2B       LD      (HL),'+'      ;'+' in Puffer
107E  F2 85 10    JP      P,1085H       ;Exponent > 0
1081  36 2D       LD      (HL),'-'      ;'-' in Puffer
1083  2F          CPL                   ;Vorzeichen entfernen
1084  3C          INC     A
1085  06 2F       LD      B,2FH         ;Ziffer = '0' - 1
1087  04          INC     B             ;Ziffer + 1 (ergibt 1. Ziffer)
1088  D6 0A       SUB     10            ;Exponent - 10 = Unterlauf?
108A  30 FB       JR      NC,1087H      ;nein, weiter
```

```
108C  C6 3A          ADD   A,3AH        ;ja, letzte Subtraktion rückgängig
                                        ;machen. + '0' ergibt 2. Ziffer
108E  23             INC   HL           ;Pufferzeiger + 1
108F  70             LD    (HL),B       ;1. Ziffer in Puffer
1090  23             INC   HL           ;Pufferzeiger + 1
1091  77             LD    (HL),A       ;2. Ziffer in Puffer
1092  23             INC   HL           ;Pufferzeiger + 1
1093  36 00          LD    (HL),0       ;Endekennung in Puffer
1095  EB             EX    DE,HL        ;Pufferendadresse in DE
1096  21 30 79       LD    HL,7930H     ;Pufferanfangsadresse in HL
1099  C9             RET                ;fertig !!!

                     Formatierten String erzeugen
109A  23             INC   HL           ;Pufferzeiger + 1
109B  C5             PUSH  BC           ;Längen-Parameter auf Stack
109C  FE 04          CP    4            ;einfache o. doppelte Genauigkeit?
109E  7A             LD    A,D          ;Format-Flag in A
109F  D2 09 11       JP    NC,1109H     ;ja!

                     Integer in String umwandeln
10A2  1F             RRA                ;Exponenten-Ausgabe ? (Bit 0)
10A3  DA A3 11       JP    C,11A3H      ;ja!
10A6  01 03 06       LD    BC,0603H     ;Parameter für '.' und ','
                                        ;'.' nach 5.Stelle, ',' nach 2. St.
10A9  CD 89 12       CALL  1289H        ;keine ','-Ausgabe ?
10AC  D1             POP   DE           ;Längen-Parameter in DE laden
10AD  7A             LD    A,D          ;Vorkommastellen - 5 >= 0 ?
10AE  D6 05          SUB   5
10B0  F4 69 12       CALL  P,1269H      ;entspr. Anzahl Nullen ausgeben
10B3  CD 2F 13       CALL  132FH        ;Zahl in String umwandeln
10B6  7B             LD    A,E          ;keine Nachkommastellen?
10B7  B7             OR    A
10B8  CC 2F 09       CALL  Z,092FH      ;ja, '.' in Puffer löschen
10BB  3D             DEC   A            ;Nachkommastellen - 1 > 0 ?
10BC  F4 69 12       CALL  P,1269H      ;entspr. Anzahl Nullen ausgeben
10BF  E5             PUSH  HL           ;Pufferzeiger auf Stack

                     Restliche Formatierung
                     Richtige Feldlänge herstellen
10C0  CD F5 0F       CALL  0FF5H        ;Restliche Formatvorschriften erl.
                                        ;Vorzeichen hinter Zahl ?
10C3  E1             POP   HL           ;Pufferzeiger laden
10C4  28 02          JR    Z,10C8H      ;nein!
10C6  70             LD    (HL),B       ;Vorzeichen hinter Zahl setzen
```

```
10C7  23          INC   HL          ;Pufferzeiger + 1
10C8  36 00       LD    (HL),0      ;Zeilenende mit X'00' markieren
10CA  21 2F 79    LD    HL,792FH    ;Adresse vor Puffer laden
10CD  23          INC   HL          ;Pufferadresse + 1
10CE  3A F3 78    LD    A,(78F3H)   ;LSB '.'-Position
10D1  95          SUB   L           ;- LSB Pufferadresse
10D2  92          SUB   D           ;- Vorkommastellen = 0 ?
10D3  C8          RET   Z           ;ja, fertig
10D4  7E          LD    A,(HL)      ;Zeichen laden
10D5  FE 20       CP    20H         ;= ' ' ?
10D7  28 F4       JR    Z,10CDH     ;ja, weiter
10D9  FE 2A       CP    2AH         ;= '*' ?
10DB  28 F0       JR    Z,10CDH     ;ja, weiter
10DD  2B          DEC   HL          ;Pufferzeiger -1
10DE  E5          PUSH  HL          ;und auf Stack
10DF  F5          PUSH  AF          ;Zeichen + Flag auf Stack
10E0  01 DF 10    LD    BC,10DFH    ;Rücksprungadresse setzen
10E3  C5          PUSH  BC
10E4  D7          RST   10H         ;nächstes Zeichen
10E5  FE 2D       CP    2DH         ;= '-' ?
10E7  C8          RET   Z           ;ja, weiter
10E8  FE 2B       CP    2BH         ;= '+' ?
10EA  C8          RET   Z           ;ja, weiter
10EB  FE 24       CP    24H         ;= '$' ?
10ED  C8          RET   Z           ;ja, weiter
10EE  C1          POP   BC          ;Rücksprungadresse wieder entfernen
10EF  FE 30       CP    30H         ;= '0' ?
10F1  20 0F       JR    NZ,1102H    ;nein, Felduberlauf
10F3  23          INC   HL          ;Pufferzeiger + 1 (hinter '.')
10F4  D7          RST   10H         ;nächstes Zeichen. =Ziffer ?
10F5  30 0B       JR    NC,1102H    ;nein, Felduberlauf
10F7  2B          DEC   HL          ;Pufferzeiger auf '.'
10F8  01          DEFB  01          ;LD BC,772B  Dummy-Befehl
10F9  2B          DEC   HL          ;Pufferzeiger - 1
10FA  77          LD    (HL),A      ;Zeichen in Puffer
10FB  F1          POP   AF          ;Zeichen vom Stack holen
10FC  28 FB       JR    Z,10F9H     ;letztes Zeichen? nein-zu 10F9H
10FE  C1          POP   BC          ;Pufferzeiger vom Stack holen
10FF  C3 CE 10    JP    10CEH       ;weiter bei 10CEH

                  Felduberlauf
1102  F1          POP   AF          ;Zeichen vom Stack holen
1103  28 FD       JR    Z,1102H     ;letztes Zeichen?
1105  E1          POP   HL          ;Pufferzeiger laden
```

```
1106  36 25          LD      (HL),25         ;'%' für Feldüberlauf vor Zahl
1108  C9             RET

                     Formatierten String von Zahlen einfacher oder
                     doppelter Genauigkeit erzeugen
1109  E5             PUSH    HL              ;Pufferzeiger auf Stack
110A  1F             RRA                     ;Exponentenausgabe ?
110B  DA AA 11       JP      C,11AAH         ;ja!
110E  28 14          JR      Z,1124H         ;bei einfacher Genauigk. => Sprung
1110  11 84 13       LD      DE,1384H        ;Konstante 1D16 adressieren
1113  CD 49 0A       CALL    0A49H           ;Zahl >= 1D16?
1116  16 10          LD      D,16            ;Genauigkeit (16 Stellen) in D
1118  FA 32 11       JP      M,1132H         ;Zahl < 1D16!

                     Feldüberlauf
111B  E1             POP     HL              ;Pufferzeiger laden
111C  C1             POP     BC              ;Längen-Parameter laden
111D  CD BD 0F       CALL    0FBDH           ;unformatierten String erzeugen
1120  2B             DEC     HL              ;Pufferzeiger - 1
1121  36 25          LD      (HL),25H        ;'%' für Feldüberlauf vor String
1123  C9             RET

                     Zahl einfacher Genauigkeit
1124  01 0E B6       LD      BC,0B60EH       ;Y = 1E6 setzen
1127  11 CA 1B       LD      DE,1BCAH
112A  CD 0C 0A       CALL    0A0CH           ;Zahl >= 1E6 ?
112D  F2 1B 11       JP      P,111BH         ;ja, Feldüberlauf
1130  16 06          LD      D,6             ;Genauigkeit (6 Stellen) in D
1132  CD 55 09       CALL    0955H           ;Zahl = 0 ?
1135  C4 01 12       CALL    NZ,1201H        ;nein, Exp. - Genauigk. + 1 in A
1138  E1             POP     HL              ;Pufferzeiger laden
1139  C1             POP     BC              ;Längen-Parameter laden
113A  FA 57 11       JP      M,1157H         ;Nachkommastellen ? ja - Sprung

                     keine Nachkommastellen
113D  C5             PUSH    BC              ;Längenparameter auf Stack
113E  5F             LD      E,A             ;Exp. - Genauigkeit + 1 in E
113F  78             LD      A,B             ;Vorkommafeldlänge in A
1140  92             SUB     D               ;- Exponent
1141  93             SUB     E               ;- 1 >= 0 ?
1142  F4 69 12       CALL    P,1269H         ;ja, entspr.Anzahl Nullen in Puffer
1145  CD 7D 12       CALL    127DH           ;Parameter f. '.' u. ',' ermitteln
1148  CD A4 12       CALL    12A4H           ;String erzeugen
114B  B3             OR      E               ;Exponent-Genauigkeit+1 > 0 ?
```

```
114C  C4 77 12        CALL    NZ,1277H        ;ja, entspr.Anzahl Nullen in Puffer
                                              ;mit '.' und ','
114F  B3              OR      E               ;Exponent-Genauigkeit+1 > 0 ?
1150  C4 91 12        CALL    NZ,1291H        ;ja, '.' und ',' in Puffer
1153  D1              POP     DE              ;Längen-Parameter laden
1154  C3 B6 10        JP      10B6H           ;restl. Formatierung ausführen

                      Nachkommastellen vorhanden
1157  5F              LD      E,A             ;Exponent - Genauigkeit + 1 nach E
1158  79              LD      A,C             ;Nachkommafeldlänge in A
1159  B7              OR      A               ;> 0 ?
115A  C4 16 0F        CALL    NZ,0F16H        ;ja, - 1 für '.'
115D  83              ADD     A,E             ;mehr als vorhanden ?
115E  FA 62 11        JP      M,1162H         ;ja!
1161  AF              XOR     A               ;nein, Anzahl überfl.Stellen = 0
1162  C5              PUSH    BC              ;Längen-Parameter auf Stack
1163  F5              PUSH    AF              ;-Anzahl überfl.Stellen auf Stack
1164  FC 18 0F        CALL    M,0F18H         ;überflüssige Stellen entfernen
1167  FA 64 11        JP      M,1164H         ;fertig? nein-zurück
116A  C1              POP     BC              ;-Anzahl überfl.Stellen vom Stack
116B  7B              LD      A,E             ;-Anzahl der tats.auszugebenden
116C  90              SUB     B               ;Nachkommastellen in A
116D  C1              POP     BC              ;Längen-Parameter wieder laden
116E  5F              LD      E,A             ;Nachkommastellen in E
116F  82              ADD     A,D             ;+ Genauigkeit > 0 ?
1170  78              LD      A,B             ;Vorkommafeldlänge in A
1171  FA 7F 11        JP      M,117F          ;ja!
1174  92              SUB     D               ;Vorkommafeldlänge - Genauigkeit
1175  93              SUB     E               ;+ auszug. Nachkmmastellen > 0 ?
1176  F4 69 12        CALL    P,1269H         ;entspr. Anzahl Nullen in Puffer
1179  C5              PUSH    BC              ;Längen-Parameter auf Stack
117A  CD 7D 12        CALL    127DH           ;Parameter f. '.' u. ',' ermitteln
117D  18 11           JR      1190H           ;weiter bei 1190H

                      nur Nachkommastellen
117F  CD 69 12        CALL    1269H           ;f.Vorkommastellen Nullen in Puffer
1182  79              LD      A,C             ;Nachkommafeldlänge in A
1183  CD 94 12        CALL    1294H           ;'.' in Puffer
1186  4F              LD      C,A             ;Nachkommafeldlänge in C
1187  AF              XOR     A               ;tats.auszugebende Nachkommastellen
1188  92              SUB     D               ;- Genauigkeit
1189  93              SUB     E               ;= Anzahl der einzufügenden Nullen
118A  CD 69 12        CALL    1269H           ;Nullen in Puffer eintragen
118D  C5              PUSH    BC              ;Längen-Parameter auf Stack sichern
```

```
118E  47          LD    B,A          ;Param. für '.' u. ',' löschen
118F  4F          LD    C,A
1190  CD A4 12     CALL  12A4H        ;String in Puffer
1193  C1          POP   BC           ;Längen-Parameter vom Stack laden
1194  B1          OR    C            ;Nachkommafeldlänge > 0 ?
1195  20 03       JR    NZ,119AH     ;ja!
1197  2A F3 78    LD    HL,(78F3H)   ;'.' - Adresse laden
119A  83          ADD   A,E          ;Nachkommafeldlänge - Anzahl tats.
                                     ;ausgegebener Nachkommastellen
119B  3D          DEC   A            ;- 1 für '.'
119C  FA 69 12    CALL  P,1269H      ;> 0 ? Entspr.Anzahl Nullen ausgeb.
119F  50          LD    D,B          ;Vorkommafeldlänge in D
11A0  C3 BF 10    JP    10BFH        ;weiter bei 10BFH

                  Formatierte Exponentenausgabe
                  Ansprung für Integer
11A3  E5          PUSH  HL           ;Pufferzeiger auf Stack
11A4  D5          PUSH  DE           ;Format-Flag auf Stack
11A5  CD CC 0A    CALL  0ACCH        ;Integer in einf.Genauigk.umwandeln
11A8  D1          POP   DE           ;Format-Flag wieder laden
11A9  AF          XOR   A            ;Flag f. einf.Genauigkeit setzen

                  Ansprung f. einfache und doppelte Genauigkeit
11AA  CA B0 11    JP    Z,11B0H      ;einf.Genauigkeit? => Sprung
11AD  1E 10       LD    E,10H        ;dopp.Genauigkeit = 16 Stellen
11AF  01          DEFB  01           ;LD BC,061E  Dummy-Befehl
11B0  1E 06       LD    E,6          ;einf.Genauigkeit = 6 Stellen
11B2  CD 55 09    CALL  0955H        ;Zahl = 0 ?
11B5  37          SCF                ;ja, Carry setzen
11B6  C4 01 12    CALL  NZ,1201H     ;nein, Exponent-Genauigkeit+1,C=0
11B9  E1          POP   HL           ;Pufferzeiger laden
11BA  C1          POP   BC           ;Längen-Parameter laden
11BB  F5          PUSH  AF           ;Exp.-Genauigkeit+1 u. Flag retten
11BC  79          LD    A,C          ;Nachkommafeldlänge = 0 ?
11BD  B7          OR    A
11BE  F5          PUSH  AF           ;Nachkommafeldlänge auf Stack
11BF  C4 16 0F    CALL  NZ,0F16H     ;nein, Nachkmmafeldlänge - 1
11C2  80          ADD   A,B          ;Vorkommafeldlänge addieren
11C3  4F          LD    C,A          ;Gesamt-Feldlänge in C
11C4  7A          LD    A,D          ;Format-Flag testen
11C5  E6 04       AND   4            ;Vorzeichen hinter Zahl? (Bit 2)
11C7  FE 01       CP    1            ;ja, 0 ins Format-Flag
11C9  9F          SBC   A,A          ;sonst -1
11CA  57          LD    D,A
```

```
11CB  81          ADD   A,C          ;Gesamtlänge -1, wenn Vorzeichen
                                     ;nicht hinter Zahl
11CC  4F          LD    C,A          ;in C
11CD  93          SUB   E            ;- Exp.-Genauigk.+1 ergibt
11CE  F5          PUSH  AF           ;-Anzahl wegzurundender Stellen
11CF  C5          PUSH  BC           ;Längen-Parameter auf Stack
11D0  FC 18 0F    CALL  M,0F18H      ;Stellen wegrunden
11D3  FA D0 11    JP    M,11D0H      ;Schleife, bis Anzahl = 0
11D6  C1          POP   BC           ;Längen-Parameter laden
11D7  F1          POP   AF           ;Anzahl weggerundeter Stellen
11D8  C5          PUSH  BC           ;Längen-Parameter wieder auf Stack
11D9  F5          PUSH  AF           ;Anzahl wegger. Stellen auf Stack
11DA  FA DE 11    JP    M,11DEH      ;Stellen weggerundet? ja zu 11DEH
11DD  AF          XOR   A            ;keine Stellen weggerundet
11DE  2F          CPL                ;positive Anzahl ermitteln
11DF  3C          INC   A            ;+ 1
11E0  80          ADD   A,B          ;+ Vorkommalänge
11E1  3C          INC   A            ;+ 1
11E2  82          ADD   A,D          ;- 1, wenn Vorzeichen vor Zahl
11E3  47          LD    B,A          ;= Position des '.'
11E4  0E 00       LD    C,0          ;Parameter f. ',' = 0 (kein ',')
11E6  CD A4 12    CALL  12A4H        ;String in Puffer übertragen
11E9  F1          POP   AF           ;Gesamtlänge - Genauigkeit > 0 ?
11EA  F4 71 12    CALL  P,1271H      ;ja, entspr.Anzahl Nullen in Puffer
11ED  C1          POP   BC           ;Längen-Parameter wieder laden
11EE  F1          POP   AF           ;Nachkommalänge = 0?
11EF  CC 2F 09    CALL  Z,092FH      ;ja, '.' in Puffer löschen
11F2  F1          POP   AF           ;Zahl = 0 ?
11F3  38 03       JR    C,11F8H      ;ja!
11F5  83          ADD   A,E          ;auszugebenden Exponenten ermitteln
11F6  90          SUB   B
11F7  92          SUB   D
11F8  C5          PUSH  BC           ;Längen-Parameter auf Stack
11F9  CD 74 10    CALL  1074H        ;Exponent in Puffer
11FC  EB          EX    DE,HL        ;Puffer-Endadresse in HL
11FD  D1          POP   DE           ;Längen-Parameter in DE
11FE  C3 BF 10    JP    10BFH        ;weiter bei 10BFH

                  Zahl so oft mit 10 multiplizieren oder durch 10
                  dividieren, bis genau 6 oder 16 Stellen vorhanden sind.
1201  D5          PUSH  DE           ;DE sichern
1202  AF          XOR   A            ;Anzahl Verschiebungen = 0
1203  F5          PUSH  AF           ;Anzahl Verschiebungen auf Stack
1204  E7          RST   20H          ;Typ testen
```

```
1205  E2 22 12      JP    PO,1222H      ;einfache Genauigkeit!
1208  3A 24 79      LD    A,(7924H)     ;Zahl >= 65536 ?
120B  FE 91         CP    91H
120D  D2 22 12      JP    NC,1222H      ;ja!
1210  11 64 13      LD    DE,1364H      ;Konstante 1D10 adressieren
1213  21 27 79      LD    HL,7927H      ;Y adressieren
1216  CD D3 09      CALL  09D3H         ;1D10 in Y übertragen
1219  CD A1 0D      CALL  0DA1H         ;Zahl * 1D10
121C  F1            POP   AF            ;Anzahl Verschiebungen laden
121D  D6 0A         SUB   10            ;-10
121F  F5            PUSH  AF            ;und wieder auf Stack
1220  18 E6         JR    1208H         ;weiter
1222  CD 4F 12      CALL  124FH         ;Zahl >=1E6 (!) oder 1D16 (#)?
                                        ;ja, weiter bei 1244H
1225  E7            RST   20H           ;Typ testen
1226  EA 34 12      JP    PE,1234H      ;dopp.Genauigkeit ? ja, zu 1234H
1229  01 43 91      LD    BC,9143H      ;Konstante 100000 in Y
122C  11 F9 4F      LD    DE,4FF9H
122F  CD 0C 0A      CALL  0A0CH         ;Zahl > 100000 ?
1232  18 06         JR    123AH         ;weiter bei 123AH
1234  11 6C 13      LD    DE,136CH      ;Konstante 1D15 adressieren
1237  CD 49 0A      CALL  0A49H         ;Zahl >= 1D15 ?
123A  F2 4C 12      JP    P,124CH       ;ja!
123D  F1            POP   AF            ;Verschiebungen laden
123E  CD 08 0F      CALL  0F08H         ;Zahl / 10, Verschiebungen + 1
1241  F5            PUSH  AF            ;Verschiebungen auf Stack
1242  18 E1         JR    1225H         ;weiter
1244  F1            POP   AF            ;Verschiebungen laden
1245  CD 18 0F      CALL  0F18H         ;Zahl * 10, Verschiebungen - 1
1248  F5            PUSH  AF            ;Verschiebungen auf Stack
1249  CD 4F 12      CALL  124FH         ;Zahl >= 1E6 (!) oder 1D16 (#) ?
                                        ;ja, weiter bei 1243H
124C  F1            POP   AF            ;Verschiebungen laden
124D  D1            POP   DE            ;DE wiederherstellen
124E  C9            RET

124F  E7            RST   20H           ;Typ testen
1250  EA 5E 12      JP    PE,125EH      ;doppelte Genauigkeit!
1253  01 74 94      LD    BC,9474H      ;Konstante 1E6 in Y
1256  11 F8 23      LD    DE,23F8H
1259  CD 0C 0A      CALL  0A0CH         ;Zahl >= 1D6 ?
125C  18 06         JR    1264H
125E  11 74 13      LD    DE,1374H      ;Konstante 1D16 adressieren
1261  CD 49 0A      CALL  0A49H         ;Zahl >= 1D16 ?
```

```
1264  E1           POP   HL            ;Rücksprungadresse laden
1265  F2 44 12     JP    P,1244H       ;ja, nach 1244H
1268  E9           JP    (HL)          ;nein, normaler Rücksprung

                   Nullen in Puffer schreiben
1269  B7           OR    A             ;Anzahl = 0 ?
126A  C8           RET   Z             ;ja, fertig
126B  3D           DEC   A             ;Anzahl - 1
126C  36 30        LD    HL,'0'        ;'0' in Puffer
126E  23           INC   HL            ;Pufferadresse + 1
126F  18 F9        JR    126AH         ;weiter bei 126AH

                   Nullen in Puffer mit '.' und ','
1271  20 04        JR    NZ,1277H      ;Anzahl > 0 ? ja, Sprung
1273  C8           RET   Z             ;= 0?, fertig
1274  CD 91 12     CALL  1291H         ;'.' und ',' setzen
1277  36 30        LD    (HL),'0'      ;'0' in Puffer
1279  23           INC   HL            ;Pufferadresse + 1
127A  3D           DEC   A             ;Anzahl - 1
127B  18 F6        JR    1273H         ;zurück

                   Parameter für '.' und ',' ermitteln
127D  7B           LD    A,E           ;Anzahl Verschiebungen in A
127E  82           ADD   A,D           ;+ Genauigkeit
127F  3C           INC   A             ;+ 1
1280  47           LD    B,A           ;= Dezimalpunkt-Stellung
1281  3C           INC   A             ;+ 1
1282  D6 03        SUB   3             ;Stellung des ',' ermitteln
1284  30 FC        JR    NC,1282H      ;-3 bis A negativ ist
1286  C6 05        ADD   A,5           ;+ 5
1288  4F           LD    C,A           ;als ','-Parameter in C
1289  3A D8 78     LD    A,(78D8H)     ;Format-Flag laden
128C  E6 40        AND   40H           ;',' gewünscht ? (Bit 6)
128E  C0           RET   NZ            ;ja, ok
128F  4F           LD    C,A           ;nein, ','-Parameter löschen
1290  C9           RET

                   '.' und ',' setzen
1291  05           DEC   B             ;Dezimalpunktstellung - 1
1292  20 08        JR    NZ,129CH      ;Dezimalpunkt erreicht ? nein!
1294  36 2E        LD    HL,'.'        ;'.' in Puffer
1296  22 F3 78     LD    (78F3H),HL    ;Adresse des '.' merken
1299  23           INC   HL            ;Pufferzeiger + 1
129A  48           LD    C,B           ;',' = 0 (kein ',' mehr setzen)
```

```
129B  C9              RET             ;fertig
129C  0D              DEC   C         ;','-Parameter - 1. Nächste Stelle?
129D  C0              RET   NZ        ;nein, zurück
129E  36 2C           LD    (HL),','  ;',' in Puffer
12A0  23              INC   HL        ;Pufferzeiger + 1
12A1  0E 03           LD    C,3       ;','-Param = 3 f. nächstes ','
12A3  C9              RET

                      Zahl einfacher und doppelter Genauigkeit
                      in ASCII - String umwandeln
12A4  D5              PUSH  DE        ;DE sichern
12A5  E7              RST   20H       ;Typ testen
12A6  E2 EA 12        JP    PO,12EAH  ;einf.Genauigkeit? weiter bei 12EAH
12A9  C5              PUSH  BC        ;Parameter f. '.' und ',' auf Stack
12AA  E5              PUSH  HL        ;Pufferzeiger auf Stack
12AB  CD FC 09        CALL  09FCH     ;Zahl in Y
12AE  21 7C 13        LD    HL,137CH  ;Konstante 0.5 adressieren
12B1  CD F7 09        CALL  09F7H     ;0.5 in X
12B4  CD 77 0C        CALL  0C77H     ;Zahl + 0.5 nach X
12B7  AF              XOR   A         ;Normalisierungs-Flag (Cy) löschen
12B8  CD 7B 0B        CALL  0B7BH     ;Nachkommastellen abtrennen
12BB  E1              POP   HL        ;Pufferzeiger laden
12BC  C1              POP   BC        ;Parameter für '.' und ',' laden
12BD  11 8C 13        LD    DE,138CH  ;Festkommakonstanten 1D15-1D16 adr.
12C0  3E 0A           LD    A,10      ;Ziffernzähler = 10
12C2  CD 91 12        CALL  1291H     ;'.' und ',' setzen
12C5  C5              PUSH  BC        ;Parameter für '.' und ',' sichern
12C6  F5              PUSH  AF        ;Ziffernzähler sichern
12C7  E5              PUSH  HL        ;Pufferzeiger sichern
12C8  D5              PUSH  DE        ;Konstanten-Adresse sichern
12C9  06 2F           LD    B,2FH     ;Ziffernwert = '0' -1
12CB  04              INC   B         ;Ziffernwert + 1
12CC  E1              POP   HL        ;Konstanten-Adresse in HL
12CD  E5              PUSH  HL        ;und wieder auf den Stack
12CE  CD 48 0D        CALL  0D48H     ;Zahl - Konstante. Unterlauf ?
12D1  30 F8           JR    NC,12CBH  ;nein, weiter
12D3  E1              POP   HL        ;Konstanten-Adresse in HL
12D4  CD 36 0D        CALL  0D36H     ;Zahl + Konstante
12D7  EB              EX    DE,HL     ;Konstantenadresse in DE (n.Konst.)
12D8  E1              POP   HL        ;Pufferzeiger laden
12D9  70              LD    (HL),B    ;Ziffer in Puffer eintragen
12DA  23              INC   HL        ;Pufferzeiger + 1
12DB  F1              POP   AF        ;Ziffernzähler laden
12DC  C1              POP   BC        ;Parameter f. '.' und ',' laden
```

```
12DD  3D          DEC   A          ;Ziffernzähler -1. 10 Ziffern erz.?
12DE  20 E2       JR    NZ,12C2H   ;nein, weiter
12E0  C5          PUSH  BC         ;Parameter f. '.' und ',' sichern
12E1  E5          PUSH  HL         ;Pufferzeiger sichern
12E2  21 1D 79    LD    HL,791DH   ;Rest (< 1D6) mit einf.Gen. in X
12E5  CD B1 09    CALL  09B1H
12E8  18 0C       JR    12F6H      ;weiter mit einfacher Genauigkeit

12EA  C5          PUSH  BC         ;Parameter f. '.' und ',' auf Stack
12EB  E5          PUSH  HL         ;Pufferzeiger auf Stack
12EC  CD 08 07    CALL  0708H      ;Zahl + 0.5 für Rndung
12EF  3C          INC   A          ;Flag f.Normalisierung löschen
12F0  CD FB 0A    CALL  0AFBH      ;Integer der Zahl in Y
12F3  CD B4 09    CALL  09B4H      ;Zahl in X eintragen
12F6  E1          POP   HL         ;Pufferzeiger laden
12F7  C1          POP   BC         ;Parameter f. '.' und ',' laden
12F8  AF          XOR   A          ;Wiederholungs-Flag löschen
12F9  11 D2 13    LD    DE,13D2H   ;Konstanten 1E5 und 1E4 adressieren
12FC  3F          CCF              ;Wiederholungs-Flag invertieren
12FD  CD 91 12    CALL  1291H      ;'.' und ',' setzen
1300  C5          PUSH  BC         ;Parameter f. '.' und ',' auf Stack
1301  F5          PUSH  AF         ;Wiederholungs-Flag auf Stack
1302  E5          PUSH  HL         ;Pufferzeiger auf Stack
1303  D5          PUSH  DE         ;Konstanten-Zeiger auf Stack
1304  CD BF 09    CALL  09BFH      ;Zahl in Y übertragen
1307  E1          POP   HL         ;Konstanten-Zeiger in HL
1308  06 2F       LD    B,2FH      ;Zifferncode = '0' - 1
130A  04          INC   B          ;Zifferncode + 1
130B  7B          LD    A,E        ;Zahl - Konstante. Unterlauf ?
130C  96          SUB   (HL)       ;LSB
130D  5F          LD    E,A
130E  23          INC   HL         ;nächste Ziffer
130F  7A          LD    A,D
1310  9E          SBC   A,(HL)
1311  57          LD    D,A
1312  23          INC   HL         ;MSB
1313  79          LD    A,C
1314  9E          SBC   A,(HL)
1315  4F          LD    C,A
1316  2B          DEC   HL         ;Konstantenzeiger - 2
1317  2B          DEC   HL         ;auf 1. Byte der Konstante
1318  30 F0       JR    NC,130AH   ;kein Unterlauf, zurück
131A  CD B7 07    CALL  07B7H      ;Zahl + Konstante
131D  23          INC   HL         ;nächste Konstante adressieren
```

```
131E  CD B4 09      CALL    09B4H           ;Zahl nach X übertragen
1321  EB            EX      DE,HL           ;Konstanten-Adresse in DE
1322  E1            POP     HL              ;Pufferzeiger laden
1323  70            LD      (HL),B          ;Ziffer in Puffer übertragen
1324  23            INC     HL              ;Pufferzeiger + 1
1325  F1            POP     AF              ;Wiederholungs-Flag laden
1326  C1            POP     BC              ;Parameter f. '.' und ',' laden
1327  38 D3         JR      C,12FCH         ;2 Durchläufe? nein-zurück
1329  13            INC     DE              ;nächste Konstante übergehen
132A  13            INC     DE
132B  3E 04         LD      A,4             ;noch 4 Ziffern in Integer-Mode
132D  18 06         JR      1335H           ;bearbeiten

                    Ansprung bei Integer
132F  D5            PUSH    DE              ;Format-Flag auf Stack
1330  11 D8 13      LD      DE,13D8H        ;Konstanten 10000 bis 1 adressieren
1333  3E 05         LD      A,5             ;Ziffernzähler = 5
1335  CD 91 12      CALL    1291H           ;'.' und ',' ausgeben
1338  C5            PUSH    BC              ;Parameter f. '.' und ',' auf Stack
1339  F5            PUSH    AF              ;Ziffernzähler auf Stack
133A  E5            PUSH    HL              ;Pufferzeiger auf Stack
133B  EB            EX      DE,HL           ;Konstanten-Adresse in HL
133C  4E            LD      C,(HL)          ;Konstante laden
133D  23            INC     HL
133E  46            LD      B,(HL)
133F  C5            PUSH    BC              ;und auf Stack retten
1340  23            INC     HL              ;nächste Konstante adressieren
1341  E3            EX      (SP),HL         ;Konstantenadresse mit Konstante
                                            ;auf Stack tauschen
1342  EB            EX      DE,HL           ;Konstantenadresse in DE
1343  2A 21 79      LD      HL,(7921H)      ;Zahl in HL laden
1346  06 2F         LD      B,2FH           ;Zifferncode = '0' - 1
1348  04            INC     B               ;Zifferncode + 1
1349  7D            LD      A,L             ;Zahl - Konstante (LSB)
134A  93            SUB     E
134B  6F            LD      L,A
134C  7C            LD      A,H             ;(MSB)
134D  9A            SBC     A,D
134E  67            LD      H,A
134F  30 F7         JR      NC,1348H        ;Unterlauf? nein-zurück
1351  19            ADD     HL,DE           ;Zahl + Konstante
1352  22 21 79      LD      (7921H),HL      ;Zahl in X speichern
1355  D1            POP     DE              ;Konstanten-Adresse laden
1356  E1            POP     HL              ;Pufferzeiger laden
```

```
1357  70              LD    (HL),B      ;Ziffer in Puffer übertragen
1358  23              INC   HL          ;Pufferzeiger + 1
1359  F1              POP   AF          ;Zifferzähler laden
135A  C1              POP   BC          ;Parameter f. '.' und ',' laden
135B  3D              DEC   A           ;Ziffernzähler - 1
135C  20 D7           JR    NZ,1335H    ;alle Ziffern? nein-zurück
135E  CD 91 12        CALL  1291H       ;'.' und ',' ausgeben
1361  77              LD    (HL),A      ;Zeilenende mit X'00' kennzeichnen
1362  D1              POP   DE          ;DE wiederherstellen
1363  C9              RET

                      ****************************************

                      Konstanten

1364  00 00 00 00 F9 02 15 A2     ;= 10 x 10 E9 (dopp. Genauigkeit)
136C  FD FF 9F 31 A9 5F 63 B2     ;= 1 x 10 E15 (dopp. Genauigkeit)
1374  FE FF 03 BF C9 1B 0E B6     ;= 1 x 10 E16 (dopp. Genauigkeit)
137C  00 00 00 00 00 00 00 80     ;= 0.5 (doppelte Genauigkeit)
                                  ;1380-1383 = 0.5 (einf.Genauigkeit)
1384  00 00 04 BF C9 1B 0E B6     ;= 1 x 10 E16 (dopp. Genauigkeit)

138C  00 80 C6 A4 7E 8D 03        ;= 1000000000000000
1393  00 40 7A 10 F3 5A 00        ;=  100000000000000
139A  00 A0 72 4E 18 09 00        ;=   10000000000000
13A1  00 10 A5 D4 E8 00 00        ;=    1000000000000
13A8  00 E8 76 48 17 00 00        ;=     100000000000
13AF  00 E4 0B 54 02 00 00        ;=      10000000000
13B6  00 CA 9A 3B 00 00 00        ;=       1000000000
13BD  00 E1 F5 05 00 00 00        ;=        100000000
13C4  80 96 98 00 00 00 00        ;=         10000000
13CB  40 42 0F 00 00 00 00        ;=          1000000

13D2  A0 86 01                    ;=           100000
13D5  10 27 00                    ;=            10000

13D8  10 27                       ;=            10000
13DA  E8 03                       ;=             1000
13DC  64 00                       ;=              100
13DE  0A 00                       ;=               10
13E0  01 00                       ;=                1

                      ****************************************
```

```
                        Unterprogramm für SQR und ATN
                        bewirkt Multiplikation mit -1
13E2  21 82 09          LD      HL,0982H        ;X = -X  - Adresse in HL
13E5  E3                EX      (SP),HL         ;mit Rücksprung-Adresse tauschen
13E6  E9                JP      (HL)            ;zur aufrufenden Routine zurück

                        ************************************

                        SQR - Funktion
                        Bildet die Wurzel einer Zahl
                        Eing.: X = Argument
                        Ausg.: X = Funktionswert
13E7  CD A4 09          CALL    09A4H           ;Argument auf den Stack packen
13EA  21 80 13          LD      HL,1380H        ;Konstante 0.5 adressieren
13ED  CD B1 09          CALL    09B1H           ;und in X übertragen
13F0  18 03             JR      13F5H           ;weiter bei 13F5H

                        ************************************

                        Potenz einer Zahl ermitteln
                        Eing.: Basis auf dem Stack
                               X = Exponent
                        Ausg.: X = Ergebnis
13F2  CD B1 0A          CALL    0AB1H           ;Exponent in einf.Genauigkeit umw.
13F5  C1                POP     BC              ;Basis in Y übertragen
13F6  D1                POP     DE
13F7  CD 55 09          CALL    0955H           ;Exponent testen
13FA  78                LD      A,B             ;Exponent der Basis in A
13FB  28 3C             JR      Z,1439H         ;Exponent = 0? ja=Ergebnis (1)
13FD  F2 04 14          JP      P,1404H         ;Exponent > 0? ja-Sprung
1400  B7                OR      A               ;Basis=0 und Exponent<0?
1401  CA 9A 19          JP      Z,199AH         ;ja, DIVISION BY ZERO - Error
1404  B7                OR      A               ;Basis=0 und Exponent>0?
1405  CA 79 07          JP      Z,0779H         ;ja, 0 als Ergebnis in X
1408  D5                PUSH    DE              ;Basis auf Stack
1409  C5                PUSH    BC
140A  79                LD      A,C             ;Basis > 0 ?
140B  F6 7F             OR      7FH
140D  CD BF 09          CALL    09BFH           ;Exponent in Y übertragen
1410  F2 21 14          JP      P,1421H         ;Basis > 0
1413  D5                PUSH    DE              ;Exponent auf Stack
1414  C5                PUSH    BC
1415  CD 40 0B          CALL    0B40H           ;Integer(Exponent) in X
1418  C1                POP     BC              ;Exponent wieder laden
```

```
1419  D1           POP   DE
141A  F5           PUSH  AF          ;LSB INT(Exponent) auf Stack
141B  CD 0C 0A     CALL  0A0CH       ;INT(Exponent) = Exponent?
141E  E1           POP   HL          ;LSB INT(Exponent) in HL
141F  7C           LD    A,H         ;INT(Exponent) ungerade?
1420  1F           RRA
1421  E1           POP   HL          ;Basis nach X übertragen
1422  22 23 79     LD    (7923H),HL  ;MSB
1425  E1           POP   HL
1426  22 21 79     LD    (7921H),HL  ;LSB
1429  DC E2 13     CALL  C,13E2H     ;Ergebnis * (-1)
142C  CC 82 09     CALL  Z,0982H     ;Basis = -Basis
142F  D5           PUSH  DE          ;Exponent auf Stack
1430  C5           PUSH  BC
1431  CD 09 08     CALL  0809H       ;LOG(Basis) nach X
1434  C1           POP   BC          ;Exponent in Y laden
1435  D1           POP   DE
1436  CD 47 08     CALL  0847H       ;LOG(Basis) * Exponent
```

```
                        ************************************

                        EXP - Funktion
                        Exponential-Funktion einer Zahl bestimmen
                        Eing.: X = Argument
                        Ausg.: X = Funktionswert
1439  CD A4 09          CALL    09A4H           ;Argument auf Stack
143C  01 38 81          LD      BC,8138H        ;Konstante 1.4427 in Y
143F  11 3B AA          LD      DE,0AA3BH
1442  CD 47 08          CALL    0847H           ;Argument / LOG(2) in X
1445  3A 24 79          LD      A,(7924H)       ;2er Exponent des Erg. > 136 ?
1448  FE 88             CP      88H
144A  D2 31 09          JP      NC,0931H        ;ja! weiter bei 0931H
144D  CD 40 0B          CALL    0B40H           ;INT(Exponent) in A und X
1450  C6 80             ADD     A,80H           ;Offset addieren
1452  C6 02             ADD     A,2             ;Exponent > 126 ?
1454  DA 31 09          JP      C,0931H         ;ja! weiter bei 0931H
1457  F5                PUSH    AF              ;Exponent (m.Offset) auf Stack
1458  21 F8 07          LD      HL,07F8H        ;INT(Arg./LOG(2)) - 1 nach X
145B  CD 0B 07          CALL    070BH
145E  CD 41 08          CALL    0841H           ;* LOG (2)
1461  F1                POP     AF              ;Exponent d. Erg. zurück
1462  C1                POP     BC              ;Argument wieder laden
1463  D1                POP     DE
1464  F5                PUSH    AF              ;Exponent wieder auf Stack
1465  CD 13 07          CALL    0713H           ;X = (LOG(2)*INT(Arg/LN2)-1)-Arg
1468  CD 82 09          CALL    0982H
146B  21 79 14          LD      HL,1479H        ;Reihe berechnen
146E  CD A9 14          CALL    14A9H
1471  11 00 00          LD      DE,0            ;0.5 * 2 ** Exponent in Y
1474  C1                POP     BC
1475  4A                LD      C,D
1476  C3 47 08          JP      0847H           ;mit Reihenergebnis multipliz.

                        Konstanten für Exponenten - Reihe

1479  08                                        ;8 Konstanten
147A  40 2E 94 74                               ;= -1.41316 E-04
147E  70 4F 2E 77                               ;=  1.32988 E-03
1482  6E 02 88 7A                               ;= -0.30136 E-03
1486  E6 A0 2A 7C                               ;=  0.0416574
148A  50 AA AA 7E                               ;= -0.166665
148E  FF FF 7F 7F                               ;=  0.5
1492  00 00 80 81                               ;= -1
```

```
1496  00 00 00 81                    ;=  1

                      ************************************

                      Reihen-Berechnung 1
                      K1 + Z + K2 * Z**3 + K3 * Z**5
                      Eing.: X = Zahl (Z)
                             HL = Adresse der Konstanten (K)
                                  (zeigt auf Anzahl-Byte)
                      Ausg.: X = Reihenergebnis
149A  CD A4 09        CALL   09A4H          ;X auf Stack übertragen
149D  11 32 0C        LD     DE,0C32H       ;Rücksprungadresse auf Stack
14A0  D5              PUSH   DE             ;(bewirkt Multipl. mit Z am Ende)
14A1  E5              PUSH   HL             ;Konstanten-Adresse auf Stack
14A2  CD BF 09        CALL   09BFH          ;Z nach Y übertragen
14A5  CD 47 08        CALL   0847H          ;Z**2 nach X
14A8  E1              POP    HL             ;Konstanten-Adresse in HL

                      ************************************

                      Reihen-Berechnung 2
                      K1 + K2 * Z + K3 * Z**2 + K4 * Z**3
                      Eing.: wie Reihen-Berechnung 1
                      Ausg.: wie Reihen-Berechnung 1
14A9  CD A4 09        CALL   09A4H          ;Z auf Stack
14AC  7E              LD     A,(HL)         ;Anzahl der Konstanten in A
14AD  23              INC    HL             ;Adresse der 1. Konstanten
14AE  CD B1 09        CALL   09B1H          ;1. Konstante in X
14B1  06              DEFB   06             ;LD B,0F1  Dummy-Befehl
14B2  F1              POP    AF             ;Konstantenzähler laden
14B3  C1              POP    BC             ;Z oder Z**2 (Reihe 2 o. 1) in Y
14B4  D1              POP    DE
14B5  3D              DEC    A              ;Konstantenzähler -1
14B6  C8              RET    Z              ;fertig!
14B7  D5              PUSH   DE             ;Y wieder auf Stack
14B8  C5              PUSH   BC
14B9  F5              PUSH   AF             ;Konstantenzähler auf Stack
14BA  E5              PUSH   HL             ;Konstanten-Adresse auf Stack
14BB  CD 47 08        CALL   0847H          ;X * Z (o. Z**2)
14BE  E1              POP    HL             ;Konstanten-Adresse laden
14BF  CD C2 09        CALL   09C2H          ;nächste Konstante in Y
14C2  E5              PUSH   HL             ;Konstanten-Adresse auf Stack
14C3  CD 16 07        CALL   0716H          ;Konstante auf X addieren
14C6  E1              POP    HL             ;Konstanten-Adresse wieder laden
```

```
14C7  18 E9           JR     14B2H          ;weiter

                      ************************************

                      RND - Funktion
                      Zufallszahl erzeugen
                      Eing.: I = Argument (0 oder Intervallende)
                      Ausg.: I = Zufallszahl
14C9  CD 7F 0A        CALL   0A7FH          ;Argument in Integer umwandeln
14CC  7C              LD     A,H            ;< 0 ?
14CD  B7              OR     A
14CE  FA 4A 1E        JP     M,1E4AH        ;ja, FUNCTION CODE - Error
14D1  B5              OR     L              ;Argument = 0 ?
14D2  CA F0 14        JP     Z,14F0H        ;ja, Zufallszahl zw. 0 und 1 erz.
14D5  E5              PUSH   HL             ;Argument auf Stack
14D6  CD F0 14        CALL   14F0H          ;Reelle Zufallszahl in X
14D9  CD BF 09        CALL   09BFH          ;in Y übertragen
14DC  EB              EX     DE,HL          ;Argument wieder laden und
14DD  E3              EX     (SP),HL        ;Zufallszahl auf Stack
14DE  C5              PUSH   BC
14DF  CD CF 0A        CALL   0ACFH          ;Argument mit einf. Gen. in X
14E2  C1              POP    BC             ;Zufallszahl wieder in Y
14E3  D1              POP    DE
14E4  CD 47 08        CALL   0847H          ;Zufallszahl * Argument
14E7  21 F8 07        LD     HL,07F8H       ;+ 1
14EA  CD 0B 07        CALL   070BH
14ED  C3 40 0B        JP     0B40H          ;Erg. = INT(Zuf.zahl * Arg. + 1)

                      Neue Zufallszahl = alte Zufallszahl * 4253261 + 372837
14F0  21 90 78        LD     HL,7890H       ;Adresse des Multiplikators
14F3  E5              PUSH   HL             ;auf Stack
14F4  11 00 00        LD     DE,0           ;Ergebnis-Reg. löschen (CDE)
14F7  4B              LD     C,E
14F8  26 03           LD     H,3            ;Bytezähler = 3
14FA  2E 08           LD     L,8            ;Bitzähler = 8
14FC  EB              EX     DE,HL          ;Ergebnis-Register * 2
14FD  29              ADD    HL,HL          ;LSB
14FE  EB              EX     DE,HL
14FF  79              LD     A,C            ;MSB
1500  17              RLA
1501  4F              LD     C,A
1502  E3              EX     (SP),HL        ;Multiplikator-Adresse in HL
1503  7E              LD     A,(HL)         ;Byte des Multiplikators in A
1504  07              RLCA                  ;höchstwertiges Bit in Carry
```

```
1505  77           LD    (HL),A        ;und wieder zurückspeichern
1506  E3           EX    (SP),HL       ;Multiplikator-Adresse auf Stack
1507  D2 16 15     JP    NC,1516H      ;Bit nicht gesetzt, keine Addition
150A  E5           PUSH  HL            ;Zähler auf Stack
150B  2A AA 78     LD    HL,(78AAH)    ;Letzte Zufallszahl addieren
150E  19           ADD   HL,DE         ;LSB
150F  EB           EX    DE,HL
1510  3A AC 78     LD    A,(78ACH)     ;MSB
1513  89           ADC   A,C
1514  4F           LD    C,A
1515  E1           POP   HL            ;Zähler wieder laden
1516  2D           DEC   L             ;Bitzähler - 1
1517  C2 FC 14     JP    NZ,14FCH      ;Byte abgearbeitet? nein-zurück
151A  E3           EX    (SP),HL       ;Multiplikator-Adresse in HL
151B  23           INC   HL            ;+ 1
151C  E3           EX    (SP),HL       ;und wieder auf den Stack
151D  25           DEC   H             ;Bytezähler - 1
151E  C2 FA 14     JP    NZ,14FAH      ;fertig? nein-zurück
1521  E1           POP   HL            ;Stack korrigieren
1522  21 65 B0     LD    HL,0B065H     ;Ergebnis+372837 = neue Zufallszahl
1525  19           ADD   HL,DE         ;LSB
1526  22 AA 78     LD    (78AAH),HL
1529  CD EF 0A     CALL  0AEFH         ;Typ = einfache Genauigkeit
152C  3E 05        LD    A,5           ;MSB
152E  89           ADC   A,C
152F  32 AC 78     LD    (78ACH),A
1532  EB           EX    DE,HL         ;in Y übertragen
1533  06 80        LD    B,80H         ;Exp. Y = 0, damit zw. 0 und 1
1535  21 25 79     LD    HL,7925H      ;Vorzeichen-Flag setzen
1538  70           LD    (HL),B        ;Ergebnis = positiv
1539  2B           DEC   HL            ;Exponent X = Exponent Y
153A  70           LD    (HL),B
153B  4F           LD    C,A           ;MSB in C
153C  06 00        LD    B,0           ;LSB löschen
153E  C3 65 07     JP    0765H         ;zur Normalisierung

                   ************************************

                   COS - Funktion
                   Ermitteln des Cosinus eines Winkels
                   Eing.: X = Argument im Bogenmaß
                   Ausg.: Y = Funktionswert
1541  21 8B 15     LD    HL,158BH      ;Konstante PI/2 adressieren
1544  CD 0B 07     CALL  070BH         ;PI/2 auf Argument addieren
```

```
                        ************************************

                        SIN - Funktion
                        Ermitteln des Sinus eines Winkels
                        Eing.: X = Argument im Bogenmaß
                        Ausg.: X = Funktionswert
1547  CD A4 09          CALL    09A4H           ;Argument auf Stack
154A  01 49 83          LD      BC,8349H        ;Konstante 2PI in Y
154D  11 DB 0F          LD      DE,0FDBH
1550  CD B4 09          CALL    09B4H           ;2PI in X übertragen
1553  C1                POP     BC              ;Argument in Y
1554  D1                POP     DE
1555  CD A2 08          CALL    08A2H           ;X = Argument / 2PI
1558  CD A4 09          CALL    09A4H           ;Argument /2PI auf Stack
155B  CD 40 0B          CALL    0B40H           ;INT (Arg./2PI) in X
155E  C1                POP     BC              ;Arg./2PI vom Stack in Y
155F  D1                POP     DE
1560  CD 13 07          CALL    0713H           ;X = Arg/2PI - INT(Arg/2PI)

                        Intervall (0..1) in Intervall (-0.25 ... 0.25) transform.
1563  21 8F 15          LD      HL,158FH        ;Konstante 0.25 adressieren
1566  CD 10 07          CALL    0710H           ;0.25 - X in X
1569  CD 55 09          CALL    0955H           ;X >= 0 ?
156C  37                SCF                     ;Flag f. Multipl. mit (-1) löschen
156D  F2 77 15          JP      P,1577H         ;ja!
1570  CD 08 07          CALL    0708H           ;0.5 + X in X
1573  CD 55 09          CALL    0955H           ;X >= 0 ?
1576  B7                OR      A               ;Flag f. Multipl. mit (-1) setzen
1577  F5                PUSH    AF              ;Flag auf Stack
1578  F4 82 09          CALL    P,0982H         ;ja! X = -X
157B  21 8F 15          LD      HL,158FH        ;Konstante 0.25 adressieren
157E  CD 0B 07          CALL    070BH           ;0.25 + X in X
1581  F1                POP     AF              ;Flag laden
1582  D4 82 09          CALL    NC,0982H        ;Carry = 0 ? ja - X = -X
1585  21 93 15          LD      HL,1593H        ;Konstanten für Reihenberechnung
1588  C3 9A 14          JP      149AH           ;Reihe berechnen

                        ************************************

                        Konstanten
158B  DB 0F 49 81                               ;= 1.5708
158F  00 00 00 7F                               ;= 0.25
```

```
                        für Sinus - Reihenberechnung
1593  05                                ;Anzahl = 5
1594  BA D7 1E 86                       ;=  39.7107
1598  64 26 99 87                       ;= -76.575
159C  58 34 23 87                       ;=  81.6022
15A0  E0 5D A5 86                       ;= -41.3417
15A4  DA 0F 49 83                       ;=   6.28319

                        ************************************

                        TAN - Funktion
                        Errechnet den Tangens eines Winkels
                        Eing.: X = Argument im Bogenmaß
                        Ausg.: X = Funktionswert
15A8  CD A4 09          CALL    09A4H           ;Argument auf Stack
15AB  CD 47 15          CALL    1547H           ;Sin (Arg) ermitteln
15AE  C1                POP     BC              ;Argument in Y
15AF  E1                POP     HL
15B0  CD A4 09          CALL    09A4H           ;Sin (Arg) auf Stack
15B3  EB                EX      DE,HL
15B4  CD B4 09          CALL    09B4H           ;Argument in X übertragen
15B7  CD 41 15          CALL    1541H           ;Cos (Arg) ermitteln
15BA  C3 A0 08          JP      08A0H           ;Tan(Arg.) = Sin(Arg.) / Cos(Arg.)

                        ************************************

                        ATN - Funktion
                        Arcus-Tangens Berechnung
                        Eing.: X = Argument
                        Ausg.: X = Winkel im Bogenmaß
15BD  CD 55 09          CALL    0955H           ;Argument < 0 ?
15C0  FC E2 13          CALL    M,13E2H         ;ja, Ergebnis * (-1)
15C3  FC 82 09          CALL    M,0982H         ;Abs(Argument) in X
15C6  3A 24 79          LD      A,(7924H)       ;Argument < 1 ?
15C9  FE 81             CP      81H
15CB  38 0C             JR      C,15D9H         ;ja!
15CD  01 00 81          LD      BC,8100H        ;nein! Y = 1
15D0  51                LD      D,C
15D1  59                LD      E,C
15D2  CD A2 08          CALL    08A2H           ;X = 1 / Argument
15D5  21 10 07          LD      HL,0710H        ;Sprungadr. zu 0710 auf Stack
15D8  E5                PUSH    HL
15D9  21 E3 15          LD      HL,15E3H        ;Konstanten für Reihe adressieren
15DC  CD 9A 14          CALL    149AH           ;Reihe berechnen
```

```
15DF  21 8B 15        LD     HL,158BH      ;Adresse von PI/2 laden
15E2  C9              RET                  ;weiter bei 0710H

                      ************************************

                      Konstanten für die Arcus-Tangens Reihe
15E3  09                                   ;Anzahl = 9
15E4  4A D7 3B 7B                          ;=  2.86623 E-03
15E8  02 6E 84 7B                          ;= -0.0161657
15EC  FE C1 2F 7C                          ;=  0.0429096
15F0  74 31 9A 7D                          ;= -0.0752896
15F4  84 3D 5A 7D                          ;=  0.106563
15F8  C8 7F 91 7E                          ;= -0.142089
15FC  E4 BB 4C 7E                          ;=  0.199936
1600  6C AA AA 7F                          ;= -0.333331
1604  00 00 00 81                          ;=  1

                      ************************************

                      Sprungtabelle für Funktionen
                      (Tokens D7 bis FA)

1608  8A 09           DEFW   098AH         ;D7 = SGN
160A  37 0B           DEFW   0B37H         ;D8 = INT
160C  77 09           DEFW   0977H         ;D9 = ABS
160E  D4 27           DEFW   27D4H         ;DA = FRE
1610  EF 2A           DEFW   2AEFH         ;DB = INP
1612  F5 27           DEFW   27F5H         ;DC = POS
1614  E7 13           DEFW   13E7H         ;DD = SQR
1616  C9 14           DEFW   14C9H         ;DE = RND
1618  09 08           DEFW   0809H         ;DF = LOG
161A  39 14           DEFW   1439H         ;E0 = EXP
161C  41 15           DEFW   1541H         ;E1 = COS
161E  47 15           DEFW   1547H         ;E2 = SIN
1620  A8 15           DEFW   15A8H         ;E3 = TAN
1622  BD 15           DEFW   15BDH         ;E4 = ATN
1624  AA 2C           DEFW   2CAAH         ;E5 = PEEK
1626  52 79           DEFW   7952H         ;E6 = CVI
1628  58 79           DEFW   7958H         ;E7 = CVS
162A  5E 79           DEFW   795EH         ;E8 = CVD
162C  61 79           DEFW   7961H         ;E9 = EOF
162E  64 79           DEFW   7964H         ;EA = LOC
1630  67 79           DEFW   7967H         ;EB = LOF
1632  6A 79           DEFW   796AH         ;EC = MKI$
```

```
1634  6D 79           DEFW   796DH        ;ED = MKS$
1636  70 79           DEFW   7970H        ;EE = MKD$
1638  7F 0A           DEFW   0A7FH        ;EF = CINT
163A  B1 0A           DEFW   0AB1H        ;F0 = CSNG
163C  DB 0A           DEFW   0ADBH        ;F1 = CDBL
163E  26 0B           DEFW   0B26H        ;F2 = FIX
1640  03 2A           DEFW   2A03H        ;F3 = LEN
1642  36 28           DEFW   2836H        ;F4 = STR$
1644  C5 2A           DEFW   2AC5H        ;F5 = VAL
1646  0F 2A           DEFW   2A0FH        ;F6 = ASC
1648  1F 2A           DEFW   2A1FH        ;F7 = CHR$
164A  61 2A           DEFW   2A61H        ;F8 = LEFT$
164C  91 2A           DEFW   2A91H        ;F9 = RIGHT$
164E  9A 2A           DEFW   2A9AH        ;FA = MID$

                      ************************************

                      Tabelle der BASIC - Schlüsselworte
                      (aufsteigend nach Token sortiert)

1650  C5              DEFB   80H+'E'      ;80 = END
1651  4E 44           DEFM   'ND'
1653  C6              DEFB   80H+'F'      ;81 = FOR
1654  4F 52           DEFM   'OR'
1656  D2              DEFB   80H+'R'      ;82 = RESET
1657  45 53 45 54     DEFM   'ESET'
165B  D3              DEFB   80H+'S'      ;83 = SET
165C  45 54           DEFM   'ET'
165E  C3              DEFB   80H+'C'      ;84 = CLS
165F  4C 53           DEFM   'LS'
1661  81              DEFB   81H          ;85 = CMD    (nicht kodiert)
1662  00 00           DEFB   0,0
1664  81              DEFB   81H          ;86 = RANDOM (nicht kodiert)
1665  00 00 00 00 00  DEFB   0,0,0,0,0
166A  CE              DEFB   80H+'N'      ;87 = NEXT
166B  45 58 54        DEFM   'EXT'
166E  C4              DEFB   80H+'D'      ;88 = DATA
166F  41 54 41        DEFM   'ATA'
1672  C9              DEFB   80H+'I'      ;89 = INPUT
1673  4E 50 55 54     DEFM   'NPUT'
1677  C4              DEFB   80H+'D'      ;8A = DIM
1678  49 4D           DEFM   'IM'
167A  D2              DEFB   80H+'R'      ;8B = READ
167B  45 41 44        DEFM   'EAD'
```

```
167E  CC              DEFB  80H+'L'       ;8C = LET
167F  45 54           DEFM  'ET'
1681  C7              DEFB  80H+'G'       ;8D = GOTO
1682  4F 54 4F        DEFM  'OTO'
1685  D2              DEFB  80H+'R'       ;8E = RUN
1686  55 4E           DEFM  'UN'
1688  C9              DEFB  80H+'I'       ;8F = IF
1689  46              DEFM  'F'
168A  D2              DEFB  80H+'R'       ;90 = RESTORE
168B  45 53 54 4F     DEFM  'ESTORE'
      52 45
1691  C7              DEFB  80H+'G'       ;91 = GOSUB
1692  4F 53 55 42     DEFM  'OSUB'
1696  D2              DEFB  80H+'R'       ;92 = RETURN
1697  45 54 55 52 4E  DEFM  'ETURN'
169C  D2              DEFB  80H+'R'       ;93 = REM
169D  45 4D           DEFM  'EM'
169F  D3              DEFB  80H+'S'       ;94 = STOP
16A0  54 4F 50        DEFM  'TOP'
16A3  C5              DEFB  80H+'E'       ;95 = ELSE
16A4  4C 53 45        DEFM  'LSE'
16A7  C3              DEFB  80H+'C'       ;96 = COPY
16A8  4F 50 59        DEFM  'OPY'
16AB  C3              DEFB  80H+'C'       ;97 = COLOR
16AC  4F 4C 4F 52     DEFM  'OLOR'
16B0  D6              DEFB  80H+'V'       ;98 = VERIFY
16B1  45 52 49 46 59  DEFM  'ERIFY'
16B6  81              DEFB  81H           ;99 = DEFINT  (nicht kodiert)
16B7  00 00 00 00 00  DEFB  0,0,0,0,0
16BC  81              DEFB  81H           ;9A = DEFSNG  (nicht kodiert)
16BD  00 00 00 00 00  DEFB  0,0,0,0,0
16C2  81              DEFB  81H           ;9B = DEFDBL  (nicht kodiert)
16C3  00 00 00 00 00  DEFB  0,0,0,0,0
16C8  C3              DEFB  80H+'C'       ;9C = CRUN
16C9  52 55 4E        DEFM  'RUN'
16CC  CD              DEFB  80H+'M'       ;9D = MODE
16CD  4F 44 45        DEFM  'ODE'
16D0  D3              DEFB  80H+'S'       ;9E = SOUND
16D1  4F 55 4E 44     DEFM  'OUND'
16D5  81              DEFB  81H           ;9F = RESUME  (nicht kodiert)
16D6  00 00 00 00 00  DEFB  0,0,0,0,0
16DB  CF              DEFB  80H+'O'       ;A0 = OUT
16DC  55 54           DEFM  'UT'
16DE  81              DEFB  81H           ;A1 = ON      (nicht kodeirt)
```

```
16DF  00                DEFB  0
16E0  81                DEFB  81H             ;A2 = OPEN    (nicht kodiert)
16E1  00 00 00          DEFB  0,0,0
16E4  81                DEFB  81H             ;A3 = FIELD   (nicht kodiert)
16E5  00 00 00 00       DEFB  0,0,0,0
16E9  81                DEFB  81H             ;A4 = GET     (nicht kodiert)
16EA  00 00             DEFB  0,0
16EC  81                DEFB  81H             ;A5 = PUT     (nicht kodiert)
16ED  00 00             DEFB  0,0
16EF  81                DEFB  81H             ;A6 = CLOSE   (nicht kodiert)
16F0  00 00 00 00       DEFB  0,0,0,0
16F4  81                DEFB  81H             ;A7 = LOAD    (nicht kodiert)
16F5  00 00 00          DEFB  0,0,0
16F8  81                DEFB  81H             ;A8 = MERGE   (nicht kodiert)
16F9  00 00 00 00       DEFB  0,0,0,0
16FD  81                DEFB  81H             ;A9 = NAME    (nicht kodiert)
16FE  00 00 00          DEFB  0,0,0
1701  81                DEFB  81H             ;AA = KILL    (nicht kodiert)
1702  00 00 00          DEFB  0,0,0
1705  81                DEFB  81H             ;AB = LSET    (nicht kodiert)
1706  00 00 00          DEFB  0,0,0
1709  81                DEFB  81H             ;AC = RSET    (nicht kodiert)
170A  00 00 00          DEFB  0,0,0
170D  81                DEFB  81H             ;AD = SAVE    (nicht kodiert)
170E  00 00 00          DEFB  0,0,0
1711  81                DEFB  81H             ;AE = SYSTEM  (nicht kodiert)
1712  00 00 00 00 00    DEFB  0,0,0,0,0
1717  CC                DEFB  80H+'L'         ;AF = LPRINT
1718  50 52 49 4E 54    DEFM  'PRINT'
171D  81                DEFB  81H             ;B0 = DEF     (nicht kodiert)
171E  00 00             DEFB  0,0
1720  D0                DEFB  80H+'P'         ;B1 = POKE
1721  4F 4B 45          DEFM  'OKE'
1724  D0                DEFB  80H+'P'         ;B2 = PRINT
1725  52 49 4E 54       DEFM  'RINT'
1729  C3                DEFB  80H+'C'         ;B3 = CONT
172A  4F 4E 54          DEFM  'ONT'
172D  CC                DEFB  80H+'L'         ;B4 = LIST
172E  49 53 54          DEFM  'IST'
1731  CC                DEFB  80H+'L'         ;B5 = LLIST
1732  4C 49 53 54       DEFM  'LIST'
1736  81                DEFB  81H             ;B6 = DELETE  (nicht kodiert)
1737  00 00 00 00 00    DEFB  0,0,0,0,0
173C  81                DEFB  81H             ;B7 = AUTO    (nicht kodiert)
```

```
173D  00 00 00          DEFB  0,0,0
1740  C3                DEFB  80H+'C'       ;B8 = CLEAR
1741  4C 45 41 52       DEFM  'LEAR'
1745  C3                DEFB  80H+'C'       ;B9 = CLOAD
1746  4C 4F 41 44       DEFM  'LOAD'
174A  C3                DEFB  80H+'C'       ;BA = CSAVE
174B  53 41 56 45       DEFM  'SAVE'
174F  CE                DEFB  80H+'N'       ;BB = NEW
1750  45 57             DEFM  'EW'
1752  D4                DEFB  80H+'T'       ;BC = TAB(
1753  41 42 28          DEFM  'AB('
1756  D4                DEFB  80H+'T'       ;BD = TO
1757  4F                DEFM  'O'
1758  81                DEFB  81H           ;BE = FN      (nicht kodiert)
1759  00                DEFB  0
175A  D5                DEFB  80H+'U'       ;BF = USING
175B  53 49 4E 47       DEFM  'SING'
175F  81                DEFB  81H           ;C0 = VARPTR  (nicht kodiert)
1760  00 00 00 00 00    DEFB  0,0,0,0,0
1765  D5                DEFB  80H+'U'       ;C1 = USR
1766  53 52             DEFM  'SR'
1768  81                DEFB  81H           ;C2 = ERL     (nicht kodiert)
1769  00 00             DEFB  0,0
176B  81                DEFB  81H           ;C3 = ERR     (nicht kodiert)
176C  00 00             DEFB  0,0
176E  81                DEFB  81H           ;C4 = STRING$ (nicht kodiert)
176F  00 00 00 00       DEFB  0,0,0,0,0,0
      00 00
1775  81                DEFB  81H           ;C5 = INSTR   (nicht kodiert)
1776  00 00 00 00       DEFB  0,0,0,0
177A  D0                DEFB  80H+'P'       ;C6 = POINT
177B  4F 49 4E 54       DEFM  'OINT'
177F  81                DEFB  81H           ;C7 = TIME$   (nicht kodiert)
1780  00 00 00 00       DEFB  0,0,0,0
1784  81                DEFB  81H           ;C8 = MEM     (nicht kodiert)
1785  00 00             DEFB  0,0
1787  C9                DEFB  80H+'I'       ;C9 = INKEY$
1788  4E 4B 45 59 24    DEFM  'NKEY$'
178D  D4                DEFB  80H+'T'       ;CA = THEN
178E  48 45 4E          DEFM  'HEN'
1791  CE                DEFB  80H+'N'       ;CB = NOT
1792  4F 54             DEFM  'OT'
1794  D3                DEFB  80H+'S'       ;CC = STEP
1795  54 45 50          DEFM  'TEP'
```

```
1798  AB          DEFB  80H+'+'       ;CD = +
1799  AD          DEFB  80H+'-'       ;CE = -
179A  AA          DEFB  80H+'*'       ;CF = *
179B  AF          DEFB  80H+'/'       ;D0 = /
179C  DE          DEFB  80H+5EH       ;D1 = Pfeil hoch (potenzieren)
179D  C1          DEFB  80H+'A'       ;D2 = AND
179E  4E 44       DEFM  'ND'
17A0  CF          DEFB  80H+'O'       ;D3 = OR
17A1  52          DEFM  'R'
17A2  BE          DEFB  80H+'>'       ;D4 = >
17A3  BD          DEFB  80H+'='       ;D5 = =
17A4  BC          DEFB  80H+'<'       ;D6 = <
17A5  D3          DEFB  80H+'S'       ;D7 = SGN
17A6  47 4E       DEFM  'GN'
17A8  C9          DEFB  80H+'I'       ;D8 = INT
17A9  4E 54       DEFM  'NT'
17AB  C1          DEFB  80H+'A'       ;D9 = ABS
17AC  42 53       DEFM  'BS'
17AE  81          DEFB  81H           ;DA = FRE     (nicht kodiert)
17AF  00 00       DEFB  0,0
17B1  C9          DEFB  80H+'I'       ;DB = INP
17B2  4E 50       DEFM  'NP'
17B4  81          DEFB  81H           ;DC = POS     (nicht kodiert)
17B5  00 00       DEFB  0,0
17B7  D3          DEFB  80H+'S'       ;DD = SQR
17B8  51 52       DEFM  'QR'
17BA  D2          DEFB  80H+'R'       ;DE = RND
17BB  4E 44       DEFM  'ND'
17BD  CC          DEFB  80H+'L'       ;DF = LOG
17BE  4F 47       DEFM  'OG'
17C0  C5          DEFB  80H+'E'       ;E0 = EXP
17C1  58 50       DEFM  'XP'
17C3  C3          DEFB  80H+'C'       ;E1 = COS
17C4  4F 53       DEFM  'OS'
17C6  D3          DEFB  80H+'S'       ;E2 = SIN
17C7  49 4E       DEFM  'IN'
17C9  D4          DEFB  80H+'T'       ;E3 = TAN
17CA  41 4E       DEFM  'AN'
17CC  C1          DEFB  80H+'A'       ;E4 = ATN
17CD  54 4E       DEFM  'TN'
17CF  D0          DEFB  80H+'P'       ;E5 = PEEK
17D0  45 45 4B    DEFM  'EEK'
17D3  81          DEFB  81H           ;E6 = CVI     (nicht kodiert)
17D4  00 00       DEFB  0,0
```

```
17D6  81              DEFB  81H          ;E7 = CVS     (nicht kodiert)
17D7  00 00           DEFB  0,0
17D9  81              DEFB  81H          ;E8 = CVD     (nicht kodiert)
17DA  00 00           DEFB  0,0
17DC  81              DEFB  81H          ;E9 = EOF     (nicht kodiert)
17DD  00 00           DEFB  0,0
17DF  81              DEFB  81H          ;EA = LOC     (nicht kodiert)
17E0  00 00           DEFB  0,0
17E2  81              DEFB  81H          ;EB = LOF     (nicht kodiert)
17E3  00 00           DEFB  0,0
17E5  81              DEFB  81H          ;EC = MKI$    (nicht kodiert)
17E6  00 00 00        DEFB  0,0,0
17E9  81              DEFB  81H          ;ED = MKS$    (nicht kodiert)
17EA  00 00 00        DEFB  0,0,0
17ED  81              DEFB  81H          ;EE = MKD$    (nicht kodiert)
17EE  00 00 00        DEFB  0,0,0
17F1  81              DEFB  81H          ;EF = CINT    (nicht kodiert)
17F2  00 00 00        DEFB  0,0,0
17F5  81              DEFB  81H          ;F0 = CSNG    (nicht kodiert)
17F6  00 00 00        DEFB  0,0,0
17F9  81              DEFB  81H          ;F1 = CDBL    (nicht kodiert)
17FA  00 00 00        DEFB  0,0,0
17FD  81              DEFB  81H          ;F2 = FIX     (nicht kodiert)
17FE  00 00           DEFB  0,0
1800  CC              DEFB  80H+'L'      ;F3 = LEN
1801  45 4E           DEFM  'EN'
1803  D3              DEFB  80H+'S'      ;F4 = STR$
1804  54 52 24        DEFM  'TR$'
1807  D6              DEFB  80H+'V'      ;F5 = VAL
1808  41 4C           DEFM  'AL'
180A  C1              DEFB  80H+'A'      ;F6 = ASC
180B  53 43           DEFM  'SC'
180D  C3              DEFB  80H+'C'      ;F7 = CHR$
180E  48 52 24        DEFM  'HR$'
1811  CC              DEFB  80H+'L'      ;F8 = LEFT$
1812  45 46 54 24     DEFM  'EFT$'
1816  D2              DEFB  80H+'R'      ;F9 = RIGHT$
1817  49 47 48 54 24  DEFM  'IGHT$'
181C  CD              DEFB  80H+'M'      ;FA = MID$
181D  49 44 24        DEFM  'ID$'
1820  A7              DEFB  80H+27H      ;FB = '
1821  00              DEFB  00H          ;Ende der Tabelle
```

```
                        ************************************

                        Sprungtabelle für Befehle (Token 80 - BB)

1822  AE 1D             DEFW  1DAEH          ;80 = END
1824  A1 1C             DEFW  1CA1H          ;81 = FOR
1826  38 01             DEFW  0138H          ;82 = RESET
1828  35 01             DEFW  0135H          ;83 = SET
182A  C9 01             DEFW  01C9H          ;84 = CLS
182C  73 79             DEFW  7973H          ;85 = CMD
182E  D3 01             DEFW  01D3H          ;86 = RANDOM
1830  B6 22             DEFW  22B6H          ;87 = NEXT
1832  05 1F             DEFW  1F05H          ;88 = DATA
1834  9A 21             DEFW  219AH          ;89 = INPUT
1836  08 26             DEFW  2608H          ;8A = DIM
1838  EF 21             DEFW  21EFH          ;8B = READ
183A  21 1F             DEFW  1F21H          ;8C = LET
183C  C2 1E             DEFW  1EC2H          ;8D = GOTO
183E  A3 1E             DEFW  1EA3H          ;8E = RUN
1840  39 20             DEFW  2039H          ;8F = IF
1842  91 1D             DEFW  1D91H          ;90 = RESTORE
1844  B1 1E             DEFW  1EB1H          ;91 = GOSUB
1846  DE 1E             DEFW  1EDEH          ;92 = RETURN
1848  07 1F             DEFW  1F07H          ;93 = REM
184A  A9 1D             DEFW  1DA9H          ;94 = STOP
184C  07 1F             DEFW  1F07H          ;95 = ELSE
184E  12 39             DEFW  3912H          ;96 = COPY
1850  9D 38             DEFW  389DH          ;97 = COLOR
1852  38 37             DEFW  3738H          ;98 = VERIFY
1854  03 1E             DEFW  1E03H          ;99 = DEFINT
1856  06 1E             DEFW  1E06H          ;9A = DEFSNG
1858  09 1E             DEFW  1E09H          ;9B = DEFDBL
185A  2E 37             DEFW  372EH          ;9C = CRUN
185C  63 2E             DEFW  2E63H          ;9D = MODE
185E  F5 2B             DEFW  2BF5H          ;9E = SOUND
1860  AF 1F             DEFW  1FAFH          ;9F = RESUME
1862  FB 2A             DEFW  2AFBH          ;A0 = OUT
1864  6C 1F             DEFW  1F6CH          ;A1 = ON
1866  79 79             DEFW  7979H          ;A2 = OPEN
1868  7C 79             DEFW  797CH          ;A3 = FIELD
186A  7F 79             DEFW  797FH          ;A4 = GET
186C  82 79             DEFW  7982H          ;A5 = PUT
186E  85 79             DEFW  7985H          ;A6 = CLOSE
1870  88 79             DEFW  7988H          ;A7 = LOAD
```

```
1872  8B 79          DEFW   798BH          ;A8 = MERGE
1874  8E 79          DEFW   798EH          ;A9 = NAME
1876  91 79          DEFW   7991H          ;AA = KILL
1878  97 79          DEFW   7997H          ;AB = LSET
187A  9A 79          DEFW   799AH          ;AC = RSET
187C  A0 79          DEFW   79A0H          ;AD = SAVE
187E  00 00          DEFW   0              ;AE = SYSTEM
1880  67 20          DEFW   2067H          ;AF = LPRINT
1882  5B 79          DEFW   795BH          ;B0 = DEF
1884  B1 2C          DEFW   2CB1H          ;B1 = POKE
1886  6F 20          DEFW   206FH          ;B2 = PRINT
1888  E4 1D          DEFW   1DE4H          ;B3 = CONT
188A  2E 2B          DEFW   2B2EH          ;B4 = LIST
188C  29 2B          DEFW   2B29H          ;B5 = LLIST
188E  C6 2B          DEFW   2BC6H          ;B6 = DELETE
1890  08 20          DEFW   2008H          ;B7 = AUTO
1892  7A 1E          DEFW   1E7AH          ;B8 = CLEAR
1894  56 36          DEFW   3656H          ;B9 = CLOAD
1896  A9 34          DEFW   34A9H          ;BA = CSAVE
1898  49 1B          DEFW   1B49H          ;BB = NEW
                     ************************************

                     Prioritäts-Codes für Operatoren
                     Der Operator mit dem höheren Code hat Priorität
189A  79             DEFB   79H            ; +
189B  79             DEFB   79H            ; -
189C  7C             DEFB   7CH            ; *
189D  7C             DEFB   7CH            ; /
189E  7F             DEFB   7FH            ; ** (potenzieren)
189F  50             DEFB   50H            ; AND
18A0  46             DEFB   46H            ; OR

                     ************************************

                     Sprungtabelle für Typanpassung
18A1  DB 0A          DEFW   0ADBH          ;Umwandlung in doppelte Genauigkeit
18A3  00 00          DEFW   0              ;unbenutzt
18A5  7F 0A          DEFW   0A7FH          ;Umwandlung in Integer
18A7  F4 0A          DEFW   0AF4H          ;Typ auf String testen
                                           ;TYPE MISMATCH - Error wenn nicht!
18A9  B1 0A          DEFW   0AB1H          ;Umwandlung in einfache Genauigkeit

                     ************************************
```

```
                        Sprungtabelle für Grundrechenarten und Vergleich

                        Doppelte Genauigkeit
18AB  77 0C             DEFW    0C77H           ;Addition
18AD  70 0C             DEFW    0C70H           ;Subtraktion
18AF  A1 0D             DEFW    0DA1H           ;Multiplikation
18B1  E5 0D             DEFW    0DE5H           ;Division
18B3  78 0A             DEFW    0A78H           ;Potenzieren

                        Einfache Genauigkeit
18B5  16 07             DEFW    0716H           ;Addition
18B7  13 07             DEFW    0713H           ;Subtraktion
18B9  47 08             DEFW    0847H           ;Multiplikation
18BB  A2 08             DEFW    08A2H           ;Division
18BD  0C 0A             DEFW    0A0CH           ;Potenzieren

                        Integer
18BF  D2 0B             DEFW    0BD2H           ;Addition
18C1  C7 0B             DEFW    0BC7H           ;Subtraktion
18C3  F2 0B             DEFW    0BF2H           ;Multiplikation
18C5  90 24             DEFW    2490H           ;Division
18C7  39 0A             DEFW    0A39H           ;Potenzieren

                        ************************************

                        Fehler-Abkürzungen
                        aufsteigend nach Fehlercodes sortiert
                        (werden im LASER 110-310 nicht verwendet)
18C9  4E 46             DEFM    'NF'            ;NEXT WITHOUT FOR
18CB  53 4E             DEFM    'SN'            ;SYNTAX ERROR
18CD  52 47             DEFM    'RG'            ;RETURN WITHOUT GOSUB
18CF  4F 44             DEFM    'OD'            ;OUT OF DATA
18D1  46 43             DEFM    'FC'            ;ILLEGAL FUNCTION CALL
18D3  4F 56             DEFM    'OV'            ;OVERFLOW
18D5  4F 4D             DEFM    'OM'            ;OUT OF MEMORY
18D7  55 4C             DEFM    'UL'            ;UNDEFINED LINE
18D9  42 53             DEFM    'BS'            ;SUBSCRIPT OUT OF RANGE
18DB  44 44             DEFM    'DD'            ;REDIMENSIONED ARRAY
18DD  2F 30             DEFM    '/0'            ;DIVISION BY ZERO
18DF  49 44             DEFM    'ID'            ;ILLEGAL DIRECT OPERATION
18E1  54 4D             DEFM    'TM'            ;TYPE MISMATCH
18E3  4F 53             DEFM    'OS'            ;OUT OF STRING SPACE
18E5  4C 53             DEFM    'LS'            ;STRING TOO LONG
18E7  53 54             DEFM    'ST'            ;STRING FORMULA TOO COMPLEX
```

```
18E9  43 4E        DEFM   'CN'          ;CAN'T CONTINUE
18EB  4E 52        DEFM   'NR'          ;NO RESUME
18ED  52 57        DEFM   'RW'          ;RESUME WITHOUT ERROR
18EF  55 45        DEFM   'UE'          ;UNPRINTABLE ERROR
18F1  4D 4F        DEFM   'MO'          ;MISSING OPERAND
18F3  46 44        DEFM   'FD'          ;BAD FILE DATA
18F5  4C 33        DEFM   'L3'          ;DISK BASIC COMMAND

                   **********************************

                   Daten und Unterprogramme, die bei der BASIC -
                   Initialisierung ins RAM übertragen werden.

                   Unterprogramm für Division
18F7  D6 00        SUB    0             ;Subtraktion Z2 - Z1
18F9  6F           LD     L,A           ;wird vor jedem Aufruf modifiziert
18FA  7C           LD     A,H
18FB  DE 00        SBC    A,0
18FD  67           LD     H,A
18FE  78           LD     A,B
18FF  DE 00        SBC    A,0
1901  47           LD     B,A
1902  3E 00        LD     A,0
1904  C9           RET

                   System-Daten
1905  4A 1E        DEFW   1E4AH         ;USR-Startadresse
                                        ;vorbesetzt mit FUNCTION CODE -Err.
1907  40 E6 4D                          ;Multiplikator für RND

                   Unterprogramm für INP
190A  DB 00        IN     A,(0)         ;Eingabeport in A laden
190C  C9           RET

                   Unterprogramm für OUT
190D  D3 00        OUT    (0),A         ;A-Reg über Port ausgeben
190F  C9           RET

                   System-Daten
1910  00           DEFB   0             ;INKEY$-Zwischenspeicher
1911  00           DEFB   0             ;Letzter Fehlercode für ERR
1912  00           DEFB   0             ;Druckkopf-Position
1913  00           DEFB   0             ;Ausgabe-Flag
1914  40           DEFB   64            ;Zeilenlänge auf Schirm
```

```
1915  30              DEFB   48              ;letzte Tabposition auf Schirm
1916  00              DEFB   0               ;unbenutzt
1917  4C 7B           DEFW   7B4CH           ;Anfang des Stringbereichs
1919  FE FF           DEFW   0FFFEH          ;Aktuelle Zeilennummer
191B  E9 7A           DEFW   7AE9H           ;Programmtext-Anfang

                      ************************************

                      Texte
191D  20 45 52 52     DEFM   'ERROR'
      4F 52 00
1924  20 49 4E 20     DEFM   'IN'
      00
1929  52 45 41 44     DEFM   'READY'
      59 0D 00
1930  42 52 45 41     DEFM   'BREAK'
      4B 00

                      ************************************

                      Unterprogramm für FOR/NEXT und GOSUB/RETURN
                      holt Daten vom Stack zurück
1936  21 04 00        LD     HL,4            ;Stackpointer + 4 in HL
1939  39              ADD    HL,SP           ;(2 Rücksprungadr. übergehen)
193A  7E              LD     A,(HL)          ;Flag laden
193B  23              INC    HL
193C  FE 81           CP     81H             ;Daten von FOR-Schleife?
193E  C0              RET    NZ              ;nein, fertig
193F  4E              LD     C,(HL)          ;ja, Laufvariablen-Adresse laden
1940  23              INC    HL
1941  46              LD     B,(HL)
1942  23              INC    HL
1943  E5              PUSH   HL              ;Adresszeiger auf Stack
1944  69              LD     L,C             ;Laufvariablen-Adresse in HL
1945  60              LD     H,B
1946  7A              LD     A,D             ;Laufvariable angegeben?
1947  B3              OR     E
1948  EB              EX     DE,HL           ;nein, mit Adresse in DE zurück
1949  28 02           JR     Z,194DH
194B  EB              EX     DE,HL
194C  DF              RST    18H             ;ja, = gefundene Laufvariable?
194D  01 0E 00        LD     BC,14           ;14 in BC
1950  E1              POP    HL              ;Adresszeiger wieder laden
1951  C8              RET    Z               ;ja, fertig
```

```
1952  09              ADD     HL,BC           ;Zeiger auf nächste Stack-Daten
1953  18 E5           JR      193AH           ;dasselbe noch einmal

                      ************************************

                      Speicherplatz für einzufügende Programmzeile
                      oder Variable freimachen
                      Eing.: DE = Anfangsadresse des Quellblocks
                             BC = Endadresse des Quellblocks
                             HL = Zieladresse
1955  CD 6C 19        CALL    196CH           ;liegt HL noch im freien Speicher?
                                              ;nein, OUT OF MEMORY - Error
1958  C5              PUSH    BC              ;HL und BC tauschen
1959  E3              EX      (SP),HL
195A  C1              POP     BC
195B  DF              RST     18H             ;Anfang des Quellblocks erreicht?
195C  7E              LD      A,(HL)          ;1 Byte umspeichern
195D  02              LD      (BC),A
195E  C8              RET     Z               ;ja, fertig!
195F  0B              DEC     BC              ;Adresszeiger - 1
1960  2B              DEC     HL
1961  18 F8           JR      195BH           ;nächstes Byte

                      ************************************

                      Testen, ob 2*C Bytes frei sind
                      wenn nicht, OUT OF MEMORY - Error
1963  E5              PUSH    HL              ;HL auf Stack
1964  2A FD 78        LD      HL,(78FDH)      ;Anfangsadr. des freien Speichers
1967  06 00           LD      B,0             ;B=0
1969  09              ADD     HL,BC           ;C 2 mal auf HL addieren
196A  09              ADD     HL,BC
196B  3E              DEFB    0E5H            ;LD A,0E5H  Dummy-Befehl
196C  E5              PUSH    HL              ;testen, ob HL noch im freien Sp.
196D  3E C6           LD      A,0C6H          ;HL > FFC6H ?
196F  95              SUB     L
1970  6F              LD      L,A
1971  3E FF           LD      A,0FFH
1973  9C              SBC     A,H
1974  38 04           JR      C,197AH         ;ja, OUT OF MEMORY - Error
1976  67              LD      H,A             ;HL + 4A >= SP ?
1977  39              ADD     HL,SP
1978  E1              POP     HL              ;HL wiederherstellen
1979  D8              RET     C               ;nein, zurück
```

```
                        ************************************

                        Aufbereitung und Ausgabe der Fehlermeldungen

                        OUT OF MEMORY - Error
197A  1E 0C             LD      E,0CH           ;Fehlercode in E
197C  18 24             JR      19A2H           ;zur Meldungsausgabe

                        Implizites Ende
197E  2A A2 78          LD      HL,(78A2H)      ;Zeilennummer laden
1981  7C                LD      A,H             ;in Direkt-Mode ? (=FFFF)
1982  A5                AND     L
1983  3C                INC     A
1984  28 08             JR      Z,198EH         ;nein, Sprung in END
1986  3A F2 78          LD      A,(78F2H)       ;Trap-Flag gesetzt ?
1989  B7                OR      A
198A  1E 22             LD      E,22H           ;NO RESUME - Error-Code laden
198C  20 14             JR      NZ,19A2H        ;ja, zur Meldungsausgabe
198E  C3 C1 1D          JP      1D1CH           ;Sprung in END

                        SYNTAX ERROR in DATA-Zeile
1991  2A DA 78          LD      HL,(78DAH)      ;letzte DATA-Zeile
1994  22 A2 78          LD      (78A2H),HL      ;als aktuelle Zeilennummer

                        SYNTAX ERROR
1997  1E 02             LD      E,02            ;Fehler-Code in E
1999  01                DEFB    01              ;LD BC,141EH  Dummy-Befehl

                        DIVISION BY ZERO
199A  1E 14             LD      E,14H           ;Fehler-Code in E
199C  01                DEFB    01              ;LD BC,001EH  Dummy-Befehl

                        NEXT WITHOUT FOR
199D  1E 00             LD      E,0             ;Fehler-Code in E
199F  01                DEFB    01              ;LD BC,241EH  Dummy-Befehl

                        RESUME WITHOUT ERROR
19A0  1E 24             LD      E,24H           ;Fehler-Code in E

                        Fehlermeldung ausgeben
                        Eing.: Fehlercode in E
19A2  2A A2 78          LD      HL,(78A2H)      ;Aktuelle Zeilennummer laden
19A5  22 EA 78          LD      (78EAH),HL      ;als Fehler-Zeile speichern
```

```
19A8  22 EC 78        LD    (78ACH),HL
19AB  01 B4 19        LD    BC,19B4H        ;Fortsetzungsadresse laden
19AE  2A E8 78        LD    HL,(78E8H)      ;Stack-Anfangsadresse laden
19B1  C3 9A 1B        JP    1B9AH           ;Sprung in NEW, Stack initialis.
19B4  C1              POP   BC              ;Stack korrigieren
19B5  7B              LD    A,E             ;Fehlercode in A und C
19B6  4B              LD    C,E
19B7  32 9A 78        LD    (789AH),A       ;speichern
19BA  2A E6 78        LD    HL,(78E6H)      ;Programmzeiger laden
19BD  22 EE 78        LD    (78EEH),HL      ;als Fehlerzeiger speichern
19C0  EB              EX    DE,HL           ;und in DE
19C1  2A EA 78        LD    HL,(78EAH)      ;Zeilennummer = FFFF ?
19C4  7C              LD    A,H             ;(= Direkt-Mode)
19C5  A5              AND   L
19C6  3C              INC   A
19C7  28 07           JR    Z,19D0H         ;ja, keine Unterbrechungsparam.
19C9  22 F5 78        LD    (78F5H),HL      ;Fehler-ZNr für CONT speichern
19CC  EB              EX    DE,HL           ;Zeilenadresse in HL
19CD  22 F7 78        LD    (78F7H),HL      ;als CONT-Zeiger speichern
19D0  2A F0 78        LD    HL,(78F0H)      ;Adresse einer Fehlerroutine laden
19D3  7C              LD    A,H             ;= 0 ?
19D4  B5              OR    L
19D5  EB              EX    DE,HL           ;in DE
19D6  21 F2 78        LD    HL,78F2H        ;TRAP-Flag Adresse laden
19D9  28 08           JR    Z,19E3H         ;keine Fehlerroutine (TRAP)
19DB  A6              AND   (HL)            ;noch offener Fehler-TRAP
                                            ;(ohne RESUME) ?
19DC  20 05           JR    NZ,19E3H        ;ja, keine Fehlerbehandlung durchf.
19DE  35              DEC   (HL)            ;TRAP-Flag setzen
19DF  EB              EX    DE,HL           ;Adresse der Fehlerroutine in HL
19E0  C3 36 1D        JP    1D36H           ;Programm dort fortsetzen
19E3  AF              XOR   A               ;TRAP-Flag löschen
19E4  77              LD    (HL),A
19E5  59              LD    E,C             ;Fehlercode wieder in E
19E6  CD F9 20        CALL  20F9H           ;wenn erf., CR ausgeben
19E9  21 EC 3C        LD    HL,3CECH        ;Adresse der Fehlermeldungen
19EC  CD A6 79        CALL  79A6H           ;RAM-Erweiterungsausgang
19EF  57              LD    D,A             ;D = 0
19F0  3E 3F           LD    A,3FH           ;'?' ausgeben
19F2  CD 2A 03        CALL  032AH
19F5  CD D4 3C        CALL  3CD4H           ;Fehlermeldung ausgeben
19F8  00 00 00        DEFB  0,0,0,0,0,0     ;6 x NOP
      00 00 00
19FE  21 1D 19        LD    HL,191DH        ;Text 'ERROR' adressieren
```

```
1A01  E5           PUSH  HL          ;und auf Stack
1A02  2A EA 78     LD    HL,(78EAH)  ;Fehler-Zeilennummer laden
1A05  E3           EX    (SP),HL     ;mit Textadr. auf Stack tauschen
1A06  CD A7 28     CALL  28A7H       ;'ERROR' ausgeben
1A09  E1           POP   HL          ;Fehler-Zeilennummer vom Stack
1A0A  11 FE FF     LD    DE,0FFFEH   ;= 65534 ?
1A0D  DF           RST   18H
1A0E  CA 74 06     JP    Z,0674H     ;ja, neue System-Initialisierung
1A11  7C           LD    A,H         ;= 65535 ? (FFFF)
1A12  A5           AND   L           ;(Direkt-Mode)
1A13  3C           INC   A
1A14  C4 A7 0F     CALL  NZ,0FA7H    ;nein, 'IN  Zeile' ausgeben
1A17  3E           DEFB  3EH         ;LD A,0C1H  Dummy-Befehl

                   ************************************

                   BASIC - Hauptschleife

                   Ansprung entweder bei 1A18 oder 1A19

1A18  C1           POP   BC          ;Stack korrigieren
1A19  CD 8B 03     CALL  038BH       ;Ausgabe-Flag auf Bildschirm, wenn
                                     ;erforderlich, CR auf Drucker ausg.
1A1C  CD AC 79     CALL  79ACH       ;RAM-Erweiterungsausgang
1A1F  00 00 00     DEFB  0,0,0       ;3 x NOP
1A22  CD F9 20     CALL  20F9H       ;CR auf Bildschirm, wenn erf.
1A25  21 29 19     LD    HL,1929H    ;Text 'READY' adressieren
1A28  CD A7 28     CALL  28A7H       ;und ausgeben
1A2B  3A 9A 78     LD    A,(789AH)   ;ohne Bedeutung
1A2E  D6 02        SUB   2           ;  "       "
1A30  00 00 00     DEFB  0,0,0       ;3 x NOP
1A33  21 FF FF     LD    HL,0FFFFH   ;Aktuelle Zeilennmmer = FFFF setzen
1A36  22 A2 78     LD    (78A2H),HL
1A39  3A E1 78     LD    A,(78E1H)   ;AUTO-Funktion eingeschaltet?
1A3C  B7           OR    A
1A3D  28 3A        JR    Z,1A79H     ;nein, normale Eingabe

                   ************************************

                   Programmeingabe unter AUTO-Funktion

1A3F  2A E2 78     LD    HL,(78E2H)  ;nächste AUTO-Zeilennummer laden
1A42  E5           PUSH  HL          ;und auf den Stack
1A43  CD AF 0F     CALL  0FAFH       ;Zeilennummer ausgeben
```

```
1A46  3E 20        LD    A,' '          ;danach ein Leerzeichen
1A48  CD 2A 03     CALL  032AH
1A4B  D1           POP   DE             ;Zeilennummer in DE
1A4C  D5           PUSH  DE             ;und wieder auf Stack
1A4D  CD 2C 1B     CALL  1B2CH          ;Zeile im Programmtext suchen
1A50  DC 53 2E     CALL  C,2E53H        ;vorhanden! Zeile ausgeben
1A53  00           NOP
1A54  CD E3 03     CALL  03E3H          ;Zeile von der Tastatur einlesen
1A57  D1           POP   DE             ;AUTO-Zeilennummer laden
1A58  30 06        JR    NC,1A60H       ;kein BREAK, normal weiter
1A5A  AF           XOR   A              ;AUTO-Flag löschen
1A5B  32 E1 78     LD    (78E1H),A
1A5E  18 B9        JR    1A19H          ;zur Hauptschleife zurück
1A60  2A E4 78     LD    HL,(78E4H)     ;AUTO-Increment laden
1A63  19           ADD   HL,DE          ;auf AUTO-Zeilennummer addieren
1A64  38 F4        JR    C,1A5AH        ;überlauf, AUTO-Mode verlassen
1A66  D5           PUSH  DE             ;AUTO-Zeilennummer auf Stack
1A67  11 F9 FF     LD    DE,0FFF9       ;neue AUTO-Zeilennummer > 65528 ?
1A6A  DF           RST   18H            ;Vergleich HL/DE
1A6B  D1           POP   DE             ;AUTO-Zeilennummer wieder laden
1A6C  30 EC        JR    NC,1A5AH       ;> 65528! AUTO-Mode verlassen
1A6E  22 E2 78     LD    (78E2H),HL     ;neue AUTO-Zeilennummer merken
1A71  00 00        DEFB  0,0            ;2 x NOP
1A73  21 E7 79     LD    HL,79E7H       ;Ein-/Ausgabepuffer - 1 adressieren
1A76  C3 81 1A     JP    1A81H          ;Zeile analysieren und übernehmen

                   ************************************

                   Normale Programmeingabe ohne AUTO

1A79  00 00        DEFB  0,0            ;2 x NOP
1A7B  CD E3 03     CALL  03E3H          ;Zeile von Tastatur einlesen
1A7E  DA 33 1A     JP    C,1A33H        ;BREAK! zum Hauptschleifen-Anfang
1A81  D7           RST   10H            ;1. Zeichen <> ' ' suchen
1A82  3C           INC   A              ;=Zeilenende (00)?
1A83  3D           DEC   A
1A84  CA 33 1A     JP    Z,1A33H        ;ja, zum Hauptschleifen-Anfang
1A87  F5           PUSH  AF             ;Flag retten (Cy=1, wenn Ziffer)
1A88  CD 5A 1E     CALL  1E5AH          ;Zeilennummer dekodieren
1A8B  2B           DEC   HL             ;Pufferadresse zurück (hinter ZNr)
1A8C  7E           LD    A,(HL)         ;Zeichen laden
1A8D  FE 20        CP    ' '            ;= Leerzeichen?
1A8F  28 FA        JR    Z,1A8BH        ;ja, weiter zurück
1A91  23           INC   HL             ;Pufferzeiger auf 1.Zeichen n. ZNr
```

```
1A92  7E          LD    A,(HL)          ;Zeichen laden
1A93  FE 20       CP    ' '             ;= Leerzeichen?
1A95  CC C9 09    CALL  Z,09C9H         ;ja, 1. Leerz. übergehen
1A98  D5          PUSH  DE              ;Zeilennummer auf Stack
1A99  CD C0 1B    CALL  1BC0H           ;Zwischencode erzeugen
                                        ;(HL=Anfang-1, BC=Länge+5)
1A9C  D1          POP   DE              ;Zeilennummer wieder laden
1A9D  F1          POP   AF              ;Flag wieder laden
1A9E  22 E6 78    LD    (78E6H),HL      ;Anfang des Zwischencodes-1 als
                                        ;aktuellen Programmzeiger speichern
1AA1  CD B2 79    CALL  79B2H           ;RAM-Erweiterungsausgang
1AA4  D2 5A 1D    JP    NC,1D5AH        ;Direktbefehl ausführen
                                        ;1. Zeichen war keine Ziffer
1AA7  D5          PUSH  DE              ;Zeilennummer auf den Stack
1AA8  C5          PUSH  BC              ;Zeilenlänge auf den Stack
1AA9  AF          XOR   A               ;RESUME/RETURN-Flag löschen
1AAA  32 DD 78    LD    (78DDH),A
1AAD  D7          RST   10H             ;Zeile leer?
1AAE  B7          OR    A               ;ja, Zero-Flag = 1
1AAF  F5          PUSH  AF              ;Flag auf Stack sichern
1AB0  EB          EX    DE,HL           ;Nostalgie vom TRS-80 Editor
1AB1  22 EC 78    LD    (78ECH),HL
1AB4  EB          EX    DE,HL
1AB5  CD 2C 1B    CALL  1B2CH           ;Zeile im Programmtext suchen
1AB8  C5          PUSH  BC              ;Adresszeiger darauf im Stack sich.
1AB9  DC E4 2B    CALL  C,2BE4H         ;wenn gefunden, löschen
1ABC  D1          POP   DE              ;Zeilenadresse in DE
1ABD  F1          POP   AF              ;Flags wieder laden
1ABE  D5          PUSH  DE              ;Zeilenadresse wieder auf Stack
1ABF  28 27       JR    Z,1AE8H         ;bei Leerzeile zurück zum Anfang
1AC1  D1          POP   DE              ;Zeilenadresse wieder laden
1AC2  2A F9 78    LD    HL,(78F9H)      ;Programmend-Adresse laden
1AC5  E3          EX    (SP),HL         ;mit Zeilenlänge auf Stack tauschen
1AC6  C1          POP   BC              ;Programmend-Adresse in BC
1AC7  09          ADD   HL,BC           ;Endadresse+Zeilenlänge
1AC8  E5          PUSH  HL              ;=neue Endadresse. Auf Stack sich.
1AC9  CD 55 19    CALL  1955H           ;Platz für neue Zeile schaffen
1ACC  E1          POP   HL              ;neue Programmend-Adresse laden
1ACD  22 F9 78    LD    (78F9H),HL      ;und abspeichern
1AD0  EB          EX    DE,HL           ;Zeilenadresse in HL
1AD1  74          LD    (HL),H          ;irgendeinen Zeilenzeiger eintr.
1AD2  D1          POP   DE              ;Zeilennummer wieder laden
1AD3  E5          PUSH  HL              ;Zeilenadresse auf Stack
1AD4  23          INC   HL              ;Zeilenzeiger auf Nummernfeld
```

```
1AD5  23           INC   HL
1AD6  73           LD    (HL),E        ;Zeilennummer in Zeile eintragen
1AD7  23           INC   HL
1AD8  72           LD    (HL),D
1AD9  23           INC   HL            ;Zeilenzeiger auf 1.Textbyte
1ADA  EB           EX    DE,HL
1ADB  2A A7 78     LD    HL,(78A7H)    ;Ein-/Ausgabepuffer-Startadresse
1ADE  EB           EX    DE,HL         ;in DE
1ADF  1B           DEC   DE            ;- 2 = Beginn des Zwischencodes
1AE0  1B           DEC   DE
1AE1  1A           LD    A,(DE)        ;Zwischencode in Programmtext
1AE2  77           LD    (HL),A        ;übertragen
1AE3  23           INC   HL            ;Adresszeiger + 1
1AE4  13           INC   DE
1AE5  B7           OR    A             ;Zeilenende ? (00)
1AE6  20 F9        JR    NZ,1AE1H      ;nein, nächstes Byte übertragen
1AE8  D1           POP   DE            ;Zeilenanfangsadresse laden
1AE9  CD FC 1A     CALL  1AFCH         ;ab Zeilenadresse, Zeilenzeiger ern
1AEC  CD B5 79     CALL  79B5H         ;RAM-Erweiterungsausgang
1AEF  CD 5D 1B     CALL  1B5DH         ;Variablen-Tabelle und andere
                                       ;Programmdaten löschen
1AF2  CD B8 78     CALL  78B8H         ;RAM-Erweiterungsausgang
1AF5  C3 33 1A     JP    1A33H         ;zum Anfang der Hauptschleife

                   ************************************

                   Zeilenzeiger im ganzen Programmtext erneuern
1AF8  2A A4 78     LD    HL,(78A4H)    ;Programmtext-Anfang in DE
1AFB  EB           EX    DE,HL

                   ************************************

                   Zeilenzeiger teilweise erneuern
                   Eing.: DE = Zeilenadresse der Zeile, ab der die
                              die Zeilenzeiger erneuert werden sollen.
1AFC  62           LD    H,D           ;Zeilenanfangs-Adresse in HL
1AFD  6B           LD    L,E
1AFE  7E           LD    A,(HL)        ;Zeilenzeiger = 0?
1AFF  23           INC   HL            ;(Programmende?)
1B00  B6           OR    (HL)
1B01  C8           RET   Z             ;ja, fertig
1B02  23           INC   HL            ;Zeiger und Zeilennummer übergehen
1B03  23           INC   HL
1B04  23           INC   HL
```

```
1B05  AF          XOR   A           ;A = 0
1B06  BE          CP    (HL)        ;mit Byte aus Zeile vergleichen
1B07  23          INC   HL          ;Zeilenzeiger + 1
1B08  20 FC       JR    NZ,1B06H    ;kein Zeilenende, zurück
1B0A  EB          EX    DE,HL       ;Zeilenanfangsadresse in HL
                                    ;DE = Adresse der nächsten Zeile
1B0B  73          LD    (HL),E      ;Adresse n. Zeile als Zeilenzeiger
1B0C  23          INC   HL          ;abspeichern
1B0D  72          LD    (HL),D
1B0E  18 EC       JR    1AFCH       ;nächste Zeile

                  ************************************

                  Für LIST-Kommando Argumente analysieren
                  Eing.: Zero-Flag = 1, wenn kein Argument angegeben
                         HL = Programmtextadresse
                  Ausg.: BC = Adresse der 1. auszugebenden Zeile
                         Stack = 2. Zeilennummer
1B10  11 00 00    LD    DE,0        ;1. Zeilennummer = 0 setzen
1B13  D5          PUSH  DE          ;und auf Stack
1B14  28 09       JR    Z,1B1FH     ;keine Argumente, weiter
1B16  D1          POP   DE          ;0 vom Stack entfernen
1B17  CD 4F 1E    CALL  1E4FH       ;1. Zeilennummer dekodieren
1B1A  D5          PUSH  DE          ;und auf Stack packen
1B1B  28 0B       JR    Z,1B28H     ;keine weiteren Zeichen!
                                    ;2. Zeilennummer = 1. setzen
1B1D  CF          RST   8           ;folgt ein '-' ?
1B1E  CE          DEFB  0CEH        ;Token für '-'
1B1F  11 FA FF    LD    DE,0FFFAH   ;2. Zeilennummer = 65530 setzen
1B22  C4 4F 1E    CALL  NZ,1E4FH    ;weitere Zeichen? ja,
                                    ;2. Zeilennummer dekodieren
1B25  C2 97 19    JP    NZ,1997H    ;noch mehr Zeichen?
                                    ;ja, SYNTAX ERROR
1B28  EB          EX    DE,HL       ;2. Zeilennummer in HL
1B29  D1          POP   DE          ;1. Zeilennummer in DE
1B2A  E3          EX    (SP),HL     ;2. Zeilennummer auf Stack mit
                                    ;Rücksprungadresse tauschen.
1B2B  E5          PUSH  HL          ;Rücksprungadresse wieder auf Stack

                  ************************************

                  Zeile im Programmtext suchen
                  Eing.: DE = Nummer der Zeile
                  Ausg.: Zeile vorhanden: Carry = 1, Z-Flag = 1
```

```
                                        BC = Zeilenadresse
                                        HL = Adresse der n. Zeile
                                Zeile nicht vorhanden:
                                        Carry = 0, Z-Flag = 0
                                        BC = Zeilenadresse der n. Zeile
                                        HL = Adresse der übern. Zeile
                                nicht gefunden und Programmende erreicht:
                                        Carry = 0, Z-Flag = 1
                                        BC,HL = Programmendadresse + 2
1B2C  2A A4 78          LD      HL,(78A4H)      ;Programmanfangsadresse laden
1B2F  44                LD      B,H             ;Zeilenadresse in BC
1B30  4D                LD      C,L
1B31  7E                LD      A,(HL)          ;Programmende ?
1B32  23                INC     HL              ;(Zeilenzeiger = 0000)
1B33  B6                OR      (HL)
1B34  2B                DEC     HL
1B35  C8                RET     Z               ;ja, fertig!
1B36  23                INC     HL              ;Programmzeiger auf Zeilennummer
1B37  23                INC     HL
1B38  7E                LD      A,(HL)          ;Zeilennummer in HL laden
1B39  23                INC     HL
1B3A  66                LD      H,(HL)
1B3B  6F                LD      L,A
1B3C  DF                RST     18H             ;Vergleich HL/DE
                                                ;= gesuchte Zeile ?
1B3D  60                LD      H,B             ;Zeilenanfangsadresse laden
1B3E  69                LD      L,C
1B3F  7E                LD      A,(HL)          ;Zeilenzeiger laden
1B40  23                INC     HL
1B41  66                LD      H,(HL)
1B42  6F                LD      L,A
1B43  3F                CCF                     ;Carry-Flag invertieren
1B44  C8                RET     Z               ;gesuchte Zeile? ja-fertig
1B45  3F                CCF                     ;Carry-Flag wieder zurück
1B46  D0                RET     NC              ;Zeilennummer > gesuchte Zeile
1B47  18 E6             JR      1B2FH           ;nächste Zeile untersuchen

                        ************************************

                        NEW - Befehl
                        Alle Variablen und Zeiger zurücksetzen
                        (die String-Bereichsdefinition bleibt erhalten)
1B49  C0                RET     NZ              ;Parameter? ja-SYNTAX ERROR
1B4A  CD C9 01          CALL    01C9H           ;Bildschirm löschen
```

```
1B4D  2A A4 78        LD    HL,(78A4H)     ;Programmtextanfang in HL
1B50  CD F8 1D        CALL  1DF8H          ;TROFF aufrufen
1B53  32 E1 78        LD    (78E1H),A      ;AUTO-Mode löschen
1B56  77              LD    (HL),A         ;Zeilenzeiger = 0000 an Programm-
1B57  23              INC   HL             ;textanfang (Programm löschen)
1B58  77              LD    (HL),A
1B59  23              INC   HL             ;Zeiger hinter 0000
1B5A  22 F9 78        LD    (78F9H),HL     ;als Programmendadresse speichern
1B5D  2A A4 78        LD    HL,(78A4H)     ;Programmanfangsadresse laden
1B60  2B              DEC   HL             ;- 1
1B61  22 DF 78        LD    (78DFH),HL     ;als Zeiger zur Programmfortführung
                      Typcodetabelle = einfache Genauigkeit setzen
1B64  06 1A           LD    B,1AH          ;Zähler = 26
1B66  21 01 79        LD    HL,7901H       ;Tabellen-Anfangsadresse
1B69  36 04           LD    (HL),4         ;Code f. einf.Gen. eintragen
1B6B  23              INC   HL             ;nächstes Byte
1B6C  10 FB           DJNZ  1B69H          ;Zähler - 1. fertig ?

1B6E  AF              XOR   A              ;ja, TRAP-Flag löschen
1B6F  32 F2 78        LD    (78F2H),A
1B72  6F              LD    L,A            ;HL = 0
1B73  67              LD    H,A
1B74  22 F0 78        LD    (78F0H),HL     ;Adresse einer Fehlerroutine = 0
1B77  22 F7 78        LD    (78F7H),HL     ;CONT-Adresszeiger = 0
1B7A  2A B1 78        LD    HL,(78B1H)     ;BASIC-RAM Endadresse laden
1B7D  22 D6 78        LD    (78D6H),HL     ;als Stringbereichs-Zeiger speich.
                                           ;löscht alle String-Variablen
1B80  CD 91 1D        CALL  1D91H          ;RESTORE aufrufen
1B83  2A F9 78        LD    HL,(78F9H)     ;Programm-Endadresse laden
1B86  22 FB 78        LD    (78FBH),HL     ;= Endadresse der Variablen-Tabelle
1B89  22 FD 78        LD    (78FDH),HL     ;= Endadresse der Matrix-Tabelle
1B8C  CD BB 79        CALL  79BBH          ;RAM-Erweiterungsausgang
1B8F  C1              POP   BC             ;Rücksprungadresse laden
1B90  2A A0 78        LD    HL,(78A0H)     ;Adresse des String-Bereichs
1B93  2B              DEC   HL             ;- 2
1B94  2B              DEC   HL
1B95  22 E8 78        LD    (78E8H),HL     ;als Stackanfangsadresse speichern
1B98  23              INC   HL             ;+ 2
1B99  23              INC   HL
1B9A  F9              LD    SP,HL          ;in Stackpointer übertragen
1B9B  21 B5 78        LD    HL,78B5H       ;Zwischenspeicher f. Strings lösch.
1B9E  22 B3 78        LD    (78B3H),HL     ;(Anfangsadresse in Zeiger)
1BA1  CD 8B 03        CALL  038BH          ;Ausgabe-Flag auf Bildschirm,CR auf
                                           ;Drucker ausgeben, falls erforderl.
```

```
1BA4  CD 69 21     CALL   2169H          ;Endabfrage
1BA7  AF           XOR    A              ;A = 0
1BA8  67           LD     H,A            ;HL = 0
1BA9  6F           LD     L,A
1BAA  32 DC 78     LD     (78DCH),A      ;Indizierungs-Sperre aufheben
1BAD  E5           PUSH   HL             ;0 auf Stack als Endekennung
1BAE  C5           PUSH   BC             ;Rücksprungadr. wieder auf Stack
1BAF  2A DF 78     LD     HL,(78DFH)     ;Zeiger zur Programmfortführung
1BB2  C9           RET

                   ************************************

                   Fragezeichen ausgeben und eine Zeile einlesen
1BB3  3E 3F        LD     A,'?'          ;Fragezeichen ausgeben
1BB5  CD 2A 03     CALL   032AH
1BB8  3E 20        LD     A,' '          ;Leerzeichen ausgeben
1BBA  CD 2A 03     CALL   032AH
1BBD  C3 3A 05     JP     053AH          ;eine Zeile einlesen

                   ************************************

                   Zeile analysieren und Zwischencode erzeugen
                   Eing.: HL = Textanfangsadresse (Text mit 00 abgeschl.)
                   Ausg.: BC = Länge des Zwischencodes + 5
                          HL = Adresse vor Zwischencode
                               (= Ein-/Ausgabepuffer - 3)
1BC0  AF           XOR    A              ;DATA-Flag löschen
1BC1  32 B0 78     LD     (78B0H),A
1BC4  4F           LD     C,A            ;Zeichenzähler = 0
1BC5  EB           EX     DE,HL
1BC6  2A A7 78     LD     HL,(78A7H)     ;Adresse des Ein-/Ausgabepuffers
1BC9  2B           DEC    HL             ;- 2
1BCA  2B           DEC    HL
1BCB  EB           EX     DE,HL          ;in DE
1BCC  7E           LD     A,(HL)         ;Zeichen aus Textzeile laden
1BCD  FE 20        CP     ' '            ;= Leerzeichen ?
1BCF  CA 5B 1C     JP     Z,1C5BH        ;ja! direkt übertragen
1BD2  47           LD     B,A            ;Zeichen in B (als Trennzeichen)
1BD3  FE 22        CP     '"'            ;= Anführungszeichen ?
1BD5  CA 77 1C     JP     Z,1C77H        ;ja! String übertragen
1BD8  B7           OR     A              ;Zeilenende ?
1BD9  CA 7D 1C     JP     Z,1C7DH        ;ja! fertig
1BDC  3A B0 78     LD     A,(78B0H)      ;DATA-Flag laden
1BDF  B7           OR     A              ;gesetzt ?
```

```
1BE0  7E          LD    A,(HL)        ;Zeichen laden
1BE1  C2 5B 1C    JP    NZ,1C5BH      ;ja! direkt übertragen
1BE4  FE 3F       CP    '?'           ;= Fragezeichen ?
1BE6  3E B2       LD    A,0B2H        ;PRINT-Token laden
1BE8  CA 5B 1C    JP    Z,1C5BH       ;ja! in Zwischencode übertragen
1BEB  7E          LD    A,(HL)        ;Zeichen wieder laden
1BEC  FE 30       CP    '0'           ;Zeichen < '0' ?
1BEE  38 05       JR    C,1BF5H       ;ja, auf Schlüsselworte prüfen
1BF0  FE 3C       CP    '<'           ;Zeichen < '<' ?
1BF2  DA 5B 1C    JR    C,1C5BH       ;ja, direkt übernehmen

                  Text auf gültiges BASIC-Schlüsselwort prüfen
1BF5  D5          PUSH  DE            ;Zwischencode-Zeiger auf den Stack
1BF6  11 4F 16    LD    DE,164FH      ;Anfangsadresse der Schlüsselworte
1BF9  C5          PUSH  BC            ;Zeichenzähler auf Stack
1BFA  01 3D 1C    LD    BC,1C3DH      ;Rücksprungadresse setzen
1BFD  C5          PUSH  BC
1BFE  06 7F       LD    B,7FH         ;Token-Zähler = 7F setzen
1C00  7E          LD    A,(HL)        ;Zeichen aus Text laden
1C01  FE 61       CP    61H           ;Kleinbuchstabe?
1C03  38 07       JR    C,1C0CH       ;nein!
1C05  FE 7B       CP    7BH
1C07  30 03       JR    NC,1C0CH      ;nein!
1C09  E6 5F       AND   5FH           ;in Großbuchstaben umwandeln
1C0B  77          LD    (HL),A        ;Zeichen wieder in Text zurück
1C0C  4E          LD    C,(HL)        ;1. Zeichen laden
1C0D  EB          EX    DE,HL         ;Schlüsselwortzeiger in HL
1C0E  23          INC   HL            ;nächstes Schlüsselwort suchen
1C0F  B6          OR    (HL)          ;Anfang eines Schlüsselworts ?
1C10  F2 0E 1C    JP    P,1C0EH       ;nein, weiter
1C13  04          INC   B             ;Token-Zähler + 1
1C14  7E          LD    A,(HL)        ;1. Zeichen des Schlüsselworts
1C15  E6 7F       AND   7FH           ;Bit 7 löschen
1C17  C8          RET   Z             ;Ende der Schlüsselwort-Tabelle
1C18  B9          CP    C             ;= Textzeichen ?
1C19  20 F3       JR    NZ,1C0EH      ;nein, nächstes Schlüsselwort
1C1B  EB          EX    DE,HL         ;Adresszeiger vertauschen
1C1C  E5          PUSH  HL            ;Pufferzeiger auf den Stack
1C1D  13          INC   DE            ;Schlüsselwort-Zeiger + 1
1C1E  1A          LD    A,(DE)        ;n. Zeichen des Schlüsselworts
1C1F  B7          OR    A             ;neues Schlüsselwort ?
1C20  FA 39 1C    JP    M,1C39H       ;ja, Schlüsselwort erkannt
1C23  4F          LD    C,A           ;Zeichen in C
1C24  78          LD    A,B           ;Token = GOTO?
```

```
1C25  FE 8D       CP    8DH
1C27  20 02       JR    NZ,1C2BH      ;nein, weiter
1C29  D7          RST   10H           ;ja, Leerzeichen erlaubt
1C2A  2B          DEC   HL            ;Pufferzeiger ein Zeichen zurück
1C2B  23          INC   HL            ;Pufferzeiger auf nächstes Zeichen
1C2C  7E          LD    A,(HL)        ;Zeichen aus Text laden
1C2D  FE 61       CP    61H           ;Kleinbuchstabe ?
1C2F  38 02       JR    C,1C33H       ;nein!
1C31  E6 5F       AND   5FH           ;in Großbuchstaben umwandeln
1C33  B9          CP    C             ;= Buchstabe aus Schlüsselwort ?
1C34  28 E7       JR    Z,1C1DH       ;ja, weiter
1C36  E1          POP   HL            ;nein, Pufferzeiger wieder zurück
1C37  18 D3       JR    1C0CH         ;nächstes Schlüsselwort versuchen

                  Token ermittelt
1C39  48          LD    C,B           ;Token in C
1C3A  F1          POP   AF            ;Stack bereinigen
1C3B  EB          EX    DE,HL         ;Adresszeiger vertauschen
1C3C  C9          RET

                  Token oder Text in Zwischencode
1C3D  EB          EX    DE,HL         ;HL = Pufferzeiger
1C3E  79          LD    A,C           ;Zeichen oder Token in A
1C3F  C1          POP   BC            ;Zeichenzähler laden
1C40  D1          POP   DE            ;Zwischencode-Zeiger laden
1C41  EB          EX    DE,HL         ;Adresszeiger vertauschen
1C42  FE 95       CP    95H           ;= ELSE-Token ?
1C44  36 3A       LD    (HL),':'      ;':' in Zwischencode
1C46  20 02       JR    NZ,1C4AH      ;nein, ':' ignorieren
1C48  0C          INC   C             ;ja, Zeichenzähler + 1
1C49  23          INC   HL            ;Zwischencode-Zeiger hinter ':'
1C4A  FE FB       CP    0FBH          ;"'" - Token ?
1C4C  20 0C       JR    NZ,1C5AH      ;nein!
1C4E  36 3A       LD    (HL),':'      ;ja, ':' in Zwischencode
1C50  23          INC   HL            ;Zwischencode-Zeiger + 1
1C51  06 93       LD    B,93H         ;REM-Token in Zwischencode
1C53  70          LD    (HL),B
1C54  23          INC   HL            ;Zwischencode-Zeiger + 1
1C55  EB          EX    DE,HL         ;Adresszeiger vertauschen
1C56  0C          INC   C             ;Zeichenzähler + 2
1C57  0C          INC   C
1C58  18 1D       JR    1C77H         ;Restl.Text aus Puffer unverändert
                                      ;in Zwischencode übertragen
1C5A  EB          EX    DE,HL         ;Adresszeiger vertauschen
```

```
1C5B  23          INC   HL          ;Pufferzeiger + 1
1C5C  12          LD    (DE),A      ;Token oder Zeichen in Zwischencode
1C5D  13          INC   DE          ;Zwischencode-Zeiger + 1
1C5E  0C          INC   C           ;Zeichenzähler + 1
1C5F  D6 3A       SUB   ':'         ;= ':' ?
1C61  28 04       JR    Z,1C67H     ;ja, DATA-Flag löschen
1C63  FE 4E       CP    4EH         ;DATA - Token ? (88 - 3A)
1C65  20 03       JR    NZ,1C6AH    ;nein!
1C67  32 B0 78    LD    (78B0H),A   ;ja, DATA-Flag setzen
1C6A  D6 59       SUB   59H         ;REM - Token ? (93 - 3A)
1C6C  C2 CC 1B    JP    NZ,1BCCH    ;nein, zurück
1C6F  47          LD    B,A         ;0 als Trennzeichen in B
1C70  7E          LD    A,(HL)      ;Text bis Trennzeichen oder Zeilen-
                                    ;ende unverändert in Zwischencode
1C71  B7          OR    A           ;Zeilenende ?
1C72  28 09       JR    Z,1C7DH     ;ja, fertig
1C74  B8          CP    B           ;Trennzeichen ? (bei " = ")
1C75  28 E4       JR    Z,1C5BH     ;ja, zurück
1C77  23          INC   HL          ;Pufferzeiger + 1
1C78  12          LD    (DE),A      ;Zeichen in Zwischencode
1C79  0C          INC   C           ;Zeichenzähler + 1
1C7A  13          INC   DE          ;Zwischencode-Zeiger + 1
1C7B  18 F3       JR    1C70H       ;nächstes Zeichen

1C7D  21 05 00    LD    HL,5        ;HL = 5
1C80  44          LD    B,H         ;B = 0
1C81  09          ADD   HL,BC       ;Zeichenzähler + 5
1C82  44          LD    B,H         ;in BC
1C83  4D          LD    C,L
1C84  2A A7 78    LD    HL,(78A7H)  ;Anfadr. d. Ein-/Ausgabe-Puffers
1C87  2B          DEC   HL          ;- 3
1C88  2B          DEC   HL          ;= Zeiger auf Byte vor dem
1C89  2B          DEC   HL          ;Zwischencode
1C8A  12          LD    (DE),A      ;Zwischencodeende mit 3 Nullen
1C8B  13          INC   DE          ;markieren
1C8C  12          LD    (DE),A      ;(Endekennung bei Direktbefehlen)
1C8D  13          INC   DE
1C8E  12          LD    (DE),A
1C8F  C9          RET               ;das war's
```

****

Restart 18
Vergleich von HL und DE

```
                        Eing.: HL,DE = 16 Bit Integer ohne Vorzeichen
                        Ausg.: HL > DE: Z=0, Cy=0
                               HL = DE: Z=1, Cy=0, A=0
                               HL < DE: Z=0, Cy=1
1C90  7C                LD     A,H              ;MSB HL = MSB DE ?
1C91  92                SUB    D
1C92  C0                RET    NZ               ;nein, fertig
1C93  7D                LD     A,L              ;LSB HL = LSB DE ?
1C94  93                SUB    E
1C95  C9                RET

                        ************************************

                        Restart 1
                        Syntax-Prüfung
                        Eing.: HL = Adresse des zu prüfenden Bytes
                               Prüfbyte nach RST 8 - Befehl
                        Ausg.: HL = Zeichen nach dem Prüfbyte, wenn gleich.
                               bei Ungleichheit SYNTAX ERROR.
1C96  7E                LD     A,(HL)           ;Zeichen aus Zeigerposition laden
1C97  E3                EX     (SP),HL          ;Zeiger mit Rücksprungadr. tauschen
1C98  BE                CP     (HL)             ;= dem, dem Aufruf folg. Zeichen ?
1C99  23                INC    HL               ;Rücksprungadresse + 1
1C9A  E3                EX     (SP),HL          ;wieder mit Zeiger vertauschen
1C9B  CA 78 1D          JP     Z,1D78H          ;gleich, fortsetzen mit RST 10
1C9E  C3 97 19          JP     1997H            ;ungleich, SYNTAX ERROR

                        ************************************

                        FOR - Anweisung
1CA1  3E 64             LD     A,64H            ;Indizierung sperren
1CA3  32 DC 78          LD     (78DCH),A
1CA6  CD 21 1F          CALL   1F21H            ;Anfangswert in Laufvariable
1CA9  E3                EX     (SP),HL          ;Programmzeiger auf Stack
1CAA  CD 36 19          CALL   1936H            ;Schleife mit gleicher Lauf-
                                                ;variablen bereits auf dem Stack ?
1CAD  D1                POP    DE               ;Programmzeiger in DE
1CAE  20 05             JR     NZ,1CB5H         ;nein!
1CB0  09                ADD    HL,BC            ;ja, durch Stackkorrektur alle
                                                ;Schleifen bis dort löschen
1CB1  F9                LD     SP,HL            ;Stackpointer neu setzen
1CB2  22 E8 78          LD     (78E8H),HL       ;und neuen Anfangswert abspeichern
1CB5  EB                EX     DE,HL            ;Programmzeiger in HL
1CB6  0E 08             LD     C,8              ;noch mindestens 16 Byte frei ?
```

```
1CB8  CD 63 19        CALL   1963H          ;nein, OUT OF MEMORY - Error
1CBB  E5              PUSH   HL             ;Programmzeiger auf Stack
1CBC  CD 05 1F        CALL   1F05H          ;nächste Anweisung suchen
1CBF  E3              EX     (SP),HL        ;Programmzeiger auf n. Anweisung
                                            ;auf Stack, alten Zeiger laden
1CC0  E5              PUSH   HL             ;und auch wieder auf den Stack
1CC1  2A A2 78        LD     HL,(78A2H)     ;Zeilennummer laden
1CC4  E3              EX     (SP),HL        ;mit Zeiger auf Stack tauschen
1CC5  CF              RST    8              ;folgt ein 'TO' - Token ?
1CC6  BD              DEFB   0BDH
1CC7  E7              RST    20H            ;Typ der Laufvariablen testen
1CC8  CA F6 0A        JP     Z,0AF6H        ;String ? ja, TYPE MISMATCH - Error
1CCB  D2 F6 0A        JP     NC,0AF6H       ;dopp.Gen.? ja, TYPE MISMATCH - Err
1CCE  F5              PUSH   AF             ;Typ-Flag sichern
                                            ;(FF = Integer, 01 = einf.Genauigk)
1CCF  CD 37 23        CALL   2337H          ;Endwert-Ausdruck berechnen
1CD2  F1              POP    AF             ;Typ-Flag laden
1CD3  E5              PUSH   HL             ;Programmzeiger auf Stack
1CD4  F2 EC 1C        JP     P,1CECH        ;einf. Genauigkeit!
1CD7  CD 7F 0A        CALL   0A7FH          ;Integer, Endwert umwandeln
1CDA  E3              EX     (SP),HL        ;Programmzeiger in HL
                                            ;Endwert auf den Stack
1CDB  11 01 00        LD     DE,1           ;Erhöhungswert = 1
1CDE  7E              LD     A,(HL)         ;nächstes Zeichen laden
1CDF  FE CC           CP     0CH            ;= STEP - Token ?
1CE1  CC 01 2B        CALL   Z,2B01H        ;ja, Erhöhungswert auswerten und
                                            ;in Integer umwandeln (in DE)
1CE4  D5              PUSH   DE             ;Erhöhungswert auf den Stack
1CE5  E5              PUSH   HL             ;Programmzeiger retten
1CE6  EB              EX     DE,HL          ;Erhöhungswert in HL
1CE7  CD 9E 09        CALL   099EH          ;Erhöhungswert testen
1CEA  18 22           JR     1D0EH          ;weiter bei 1D0E

1CEC  CD B1 0A        CALL   0AB1H          ;Endwert in einf.Gen. umwandeln
1CEF  CD BF 09        CALL   09BFH          ;in Y übertragen
1CF2  E1              POP    HL             ;Programmzeiger wieder laden
1CF3  C5              PUSH   BC             ;Endwert auf den Stack
1CF4  D5              PUSH   DE
1CF5  01 00 81        LD     BC,8100H       ;Erhöhungswert = 1 in Y
1CF8  51              LD     D,C
1CF9  5A              LD     E,D
1CFA  7E              LD     A,(HL)         ;nächstes Zeichen laden
1CFB  FE CC           CP     0CCH           ;= STEP - Token ?
1CFD  3E 01           LD     A,1            ;Flag für positive Erhöhung setzen
```

```
1CFF  20 0E        JR    NZ,1D0FH      ;nein!
1D01  CD 38 23     CALL  2338H         ;Erhöhungswert auswerten
1D04  E5           PUSH  HL            ;Programmzeiger auf den Stack
1D05  CD B1 0A     CALL  0AB1H         ;Erhöhungswert in einf.Gen. umwand.
1D08  CD BF 09     CALL  09BFH         ;und in Y eintragen
1D0B  CD 55 09     CALL  0955H         ;Erhöhungswert testen (A=1  wenn
                                       ;positiv, A=FF  wenn negativ)
1D0E  E1           POP   HL            ;Programmzeiger laden
1D0F  C5           PUSH  BC            ;Erhöhungswert auf Stack
1D10  D5           PUSH  DE
1D11  4F           LD    C,A           ;Erhöhungs-Flag in C
1D12  E7           RST   20H           ;Typ des Erhöhungswerts testen
1D13  47           LD    B,A           ;Typ-Flag in B
                                       ;(01 = einf.Gen. , FF = Integer)
1D14  C5           PUSH  BC            ;Typ-Flag u. Erh.-Flag auf Stack
1D15  E5           PUSH  HL            ;Programmzeiger auf Stack
1D16  2A DF 78     LD    HL,(78DFH)    ;Adresse der Laufvariablen in HL
1D19  E3           EX    (SP),HL       ;mit Prog.zeiger auf Stack tauschen
1D1A  06 81        LD    B,81H         ;FOR-Token (81) in B
1D1C  C5           PUSH  BC            ;als Markierung auf den Stack
1D1D  33           INC   SP            ;LSB entfernen

                   ************************************

                   Programmausführung
                   HL muß auf ':' oder Zeilenende zeigen
1D1E  CD 58 03     CALL  0358H         ;Tastatur abfragen
1D21  B7           OR    A             ;neue Taste gedrückt?
1D22  C4 A0 1D     CALL  NZ,1DA0H      ;ja, analysieren
1D25  22 E6 78     LD    (78E6H),HL    ;Programmzeiger abspeichern
1D28  ED 73 E8 78  LD    (78E8H),SP    ;Stackpointer speichern
1D2C  7E           LD    A,(HL)        ;Zeichen laden
1D2D  FE 3A        CP    ':'           ;':'? (Mehrere Anweisungen in Zeile
1D2F  28 29        JR    Z,1D5AH       ;ja!
1D31  B7           OR    A             ;Zeilenende ?
1D32  C2 97 19     JP    NZ,1997H      ;nein, SYNTAX - ERROR
1D35  23           INC   HL            ;Programmende?
1D36  7E           LD    A,(HL)        ;(Zeilenzeiger = 0000)
1D37  23           INC   HL
1D38  B6           OR    (HL)
1D39  CA 7E 19     JP    Z,197EH       ;ja, implizites Ende
1D3C  23           INC   HL            ;Programmzeiger auf Zeilennummer
1D3D  5E           LD    E,(HL)        ;Zeilennummer in DE laden
1D3E  23           INC   HL
```

```
1D3F  56          LD    D,(HL)
1D40  EB          EX    DE,HL          ;ZNr. in HL, Prog.zeiger in DE
1D41  22 A2 78    LD    (78A2H),HL     ;Zeilennummer = aktuelle ZNr
1D44  3A 1B 79    LD    A,(791BH)      ;Ablaufverfolger eingeschaltet?
1D47  B7          OR    A              ;(TRON)
1D48  28 0F       JR    Z,1D59H        ;nein!
1D4A  D5          PUSH  DE             ;Programmzeiger auf Stack
1D4B  3E 3C       LD    A,3CH          ;'>' ausgeben
1D4D  CD 2A 03    CALL  032AH
1D50  CD AF 0F    CALL  0FAFH          ;Zeilennummer ausgeben
1D53  3E 3E       LD    A,3EH          ;'<' ausgeben
1D55  CD 2A 03    CALL  032AH
1D58  D1          POP   DE             ;Programmzeiger wieder laden
1D59  EB          EX    DE,HL          ;Programmzeiger in HL
1D5A  D7          RST   10H            ;nächstes Zeichen adressieren
1D5B  11 1E 1D    LD    DE,1D1EH       ;Rücksprungadresse auf Stack
1D5E  D5          PUSH  DE
1D5F  C8          RET   Z              ;Ende der Anweisung
1D60  D6 80       SUB   80H            ;Token ?
1D62  DA 21 1F    JP    C,1F21H        ;nein, Zuweisung ohne LET
1D65  FE 3C       CP    3CH            ;Anweisungs-Token ?
1D67  D2 E7 2A    JP    NC,2AE7H       ;nein!
1D6A  07          RLCA                 ;Token * 2 in BC
1D6B  4F          LD    C,A
1D6C  06 00       LD    B,0
1D6E  EB          EX    DE,HL          ;Programmzeiger in DE
1D6F  21 22 18    LD    HL,1822H       ;Anfang der Sprungtabelle
1D72  09          ADD   HL,BC          ;+ 2*Token = Zeiger auf Sprungadr.
1D73  4E          LD    C,(HL)         ;Sprungadresse laden
1D74  23          INC   HL
1D75  46          LD    B,(HL)
1D76  C5          PUSH  BC             ;und auf den Stack
1D77  EB          EX    DE,HL          ;Programmzeiger wieder in HL

                  ************************************

                  Restart 10
                  Nächstes Zeichen im Programmtext suchen
                  09, 0A (LF) und 20 (' ') werden übergangen
                  Eing.: HL = Programmzeiger
                  Ausg.: A = Zeichen
                         Carry = 1, wenn Ziffer
                         Z-Flag = 1, wenn Zeilen- oder Anweisungsende
1D78  23          INC   HL             ;Programmzeiger + 1
```

```
1D79  7E           LD    A,(HL)          ;Zeichen laden
1D7A  FE 3A        CP    ':'             ;< ':' ?
1D7C  D0           RET   NC              ;ja!
1D7D  FE 20        CP    ' '             ;Leerzeichen ?
1D7F  CA 78 1D     JP    Z,1D78H         ;ja, nächstes Zeichen
1D82  FE 0B        CP    0BH             ;< 0BH ?
1D84  30 05        JR    NC,1D8BH        ;nein!
1D86  FE 09        CP    09H             ;> 09H ? (schließt 09 u. 0A aus)
1D88  D2 78 1D     JP    NC,1D78H        ;ja, nächstes Zeichen
1D8B  FE 30        CP    '0'             ;Ziffer ?
1D8D  3F           CCF                   ;ja, Carry = 1
1D8E  3C           INC   A               ;Zeilenende ?
1D8F  3D           DEC   A
1D90  C9           RET                   ;fertig

                   ************************************

                   RESTORE - Anweisung
                   Zurücksetzen des DATA-Zeigers
1D91  EB           EX    DE,HL           ;Programmzeiger in DE
1D92  2A A4 78     LD    HL,(78A4H)      ;Programm-Startadresse laden
1D95  2B           DEC   HL              ; - 1
1D96  22 FF 78     LD    (78FFH),HL      ;als DATA-Zeiger ablegen
1D99  EB           EX    DE,HL           ;Programmzeiger wieder in HL
1D9A  C9           RET                   ;fertig

                   ************************************

                   Tasten-Betätigung während der Programmausführung
                   oder bei LIST analysieren
1D9B  CD 58 03     CALL  0358H           ;Taste betätigt ?
1D9E  B7           OR    A
1D9F  C8           RET   Z               ;nein!
1DA0  00 00 00 00 00 DEFB  0,0,0,0,0     ;5 x NOP
1DA5  32 99 78     LD    (7899H),A       ;Zeichen in INKEY$-Zwischenspeicher
1DA8  3D           DEC   A               ;BREAK ?
1DA9  C0           RET   NZ              ;nein, fertig!

                   ************************************

                   Programmunterbrechung durch BREAK
1DAA  3C           INC   A               ;A = 1 setzen (BREAK-Kennung)
1DAB  C3 B4 1D     JP    1DB4H           ;weiter bei END
```

```
                        ****************************************

                        END - Anweisung
                        Beenden der Programmausführung
1DAE  C0                RET   NZ              ;folgen Parameter? ja, Fehler
1DAF  F5                PUSH  AF              ;END-Flag (A=0) auf Stack
1DB0  CC BB 79          CALL  Z,79BBH         ;RAM-Erweiterungsausgang
1DB3  F1                POP   AF              ;END-Flag wieder laden
1DB4  22 E6 78          LD    (78E6H),HL      ;aktuellen Programmzeiger speichern
1DB7  21 B5 78          LD    HL,78B5H        ;Zwischenspeicher f. Strings lösch.
1DBA  22 B3 78          LD    (78B3H),HL      ;(Zeiger auf Anfang)
1DBD  21                DEFB  21H             ;LD HL,0FFF6H  Dmmy-Befehl

                        Ansprung bei BREAK in INPUT-Anweisung
1DBE  F6 FF             OR    0FFH            ;END=Flag = FF (BREAK in INPUT)
1DC0  C1                POP   BC              ;Rücksprungadresse vom Stack entf.
1DC1  2A A2 78          LD    HL,(78A2H)      ;aktuelle Zeilennummer laden
1DC4  E5                PUSH  HL              ;auf Stack
1DC5  F5                PUSH  AF              ;END-Flag auf Stack
1DC6  7D                LD    A,L             ;Zeilennummer = FFFF ?
1DC7  A4                AND   H               ;(Direkt-Mode)
1DC8  3C                INC   A
1DC9  28 09             JR    Z,1DD4H         ;ja!
1DCB  22 F5 78          LD    (78F5H),HL      ;nein, als CONT-Zeilennr. speichern
1DCE  2A E6 78          LD    HL,(78E6H)      ;aktuellen Programmzeiger
1DD1  22 F7 78          LD    (78F7H),HL      ;als CONT-Zeiger speichern
1DD4  CD 8B 03          CALL  038BH           ;Ausgabe-Flag  auf  Bildschirm.  CR
auf
                                              ;Drucker ausgeben, falls erforderl.
1DD7  CD F9 20          CALL  20F9H           ;CR auf Bildschirm, falls erforderl
1DDA  F1                POP   AF              ;END-Flag laden
1DDB  21 30 19          LD    HL,1930H        ;Text 'BREAK' adressieren
1DDE  C2 06 1A          JP    NZ,1A06H        ;wenn nicht END und nicht Direkt-
                                              ;Mode, 'BREAK IN Zeile' ausgeben
1DE1  C3 18 1A          JP    1A18H           ;zurück zur Hauptschleife

                        ****************************************

                        CONT - Anweisung
                        Nach BREAK oder Fehler Programmausführung fortsetzen
1DE4  2A F7 78          LD    HL,(78F7H)      ;CONT - Programmzeiger laden
1DE7  7C                LD    A,H             ;= 0000 ?
1DE8  B5                OR    L               ;(keine Fortsetzung möglich)
1DE9  1E 20             LD    E,20H           ;Fehlercode CAN'T CONTINUE laden
```

```
1DEB  CA A2 19      JP    Z,19A2H       ;ja, Fehlermeldung ausgeben
1DEE  EB            EX    DE,HL         ;Programmzeiger in DE
1DEF  2A F5 78      LD    HL,(78F5H)    ;CONT-Zeilennummer laden
1DF2  22 A2 78      LD    (78A2H),HL    ;als aktuelle Zeilennummer speich.
1DF5  EB            EX    DE,HL         ;Programmzeiger wieder in HL
1DF6  C9            RET                 ;Programmlauf fortsetzen

                    ************************************

                    TRON - Anweisung
                    Ablaufverfolger einschalten
1DF7  3E            DEFB  3EH           ;LD A,0AFH  bei TRON A<>0 setzen

                    TROFF - Anweisung
                    Ablaufverfolger ausschalten
1DF8  AF            XOR   A             ;bei TROFF  A = 0 setzen
1DF9  32 1B 79      LD    (791BH),A     ;als TRACE-Flag speichern
1DFC  C9            RET

1DFD  F1            POP   AF            ;nicht benutzt
1DFE  E1            POP   HL
1DFF  C9            RET

                    ************************************

                    DEFSTR - Anweisung
                    String-Variable definieren
1E00  1E 03         LD    E,3           ;Typcode = String in E
1E02  01            DEFB  01            ;LD BC,021EH  Dummy-Befehl

                    DEFINT - Anweisung
                    Integer-Variable definieren
1E03  1E 02         LD    E,2           ;Typcode = Integer in E
1E05  01            DEFB  01            ;LD BC,041EH  Dummy-Befehl

                    DEFSNG - Anweisung
                    Variable einfacher Genauigkeit definieren
1E06  1E 04         LD    E,4           ;Typcode = einf. Genauigkeit in E
1E08  01            DEFB  01            ;LD BC,081EH  Dummy-Befehl

                    DEFDBL - Anweisung
                    Variable doppelter Genauigkeit definieren
1E09  1E 08         LD    E,8           ;Typcode = dopp. Genauigkeit in E
```

```
                        gemeinsame Routine
1E0B  CD 3D 1E          CALL    1E3DH           ;n. Textzeichen = Buchstabe?
1E0E  01 97 19          LD      BC,1997H        ;SN-Error Routine adressieren
1E11  C5                PUSH    BC              ;und auf Stack packen
1E12  D8                RET     C               ;kein Buchstabe, SYNTAX ERROR ausg.
1E13  D6 41             SUB     41H             ;Stellung im Alphabet ermitteln
1E15  4F                LD      C,A             ;in B und C übertragen
1E16  47                LD      B,A
1E17  D7                RST     10H             ;nächstes Zeichen laden
1E18  FE CE             CP      0CEH            ;= '-'-Token
1E1A  20 09             JR      NZ,1E25H        ;nein!
1E1C  D7                RST     10H             ;nächstes Zeichen laden
1E1D  CD 3D 1E          CALL    1E3DH           ;= Buchstabe ?
1E20  D8                RET     C               ;nein, SYNTAX ERROR ausgeben
1E21  D6 41             SUB     41H             ;Stellung im Alphabet ermitteln
1E23  47                LD      B,A             ;als Ober-Wert in B
1E24  D7                RST     10H             ;nächstes Zeichen adressieren
1E25  78                LD      A,B             ;2. Buchstabe < 1. Buchstabe ?
1E26  91                SUB     C
1E27  D8                RET     C               ;ja, SYNTAX - ERROR ausgeben
1E28  3C                INC     A               ;Differenz + 1 = Zähler
1E29  E3                EX      (SP),HL         ;Programmzeiger auf Stack
                                                ;Adresse der SN-Routine löschen
1E2A  21 01 79          LD      HL,7901H        ;Typcode-Tabelle adressieren
1E2D  06 00             LD      B,0             ;Offset f. 1. Buchstaben in BC
1E2F  09                ADD     HL,BC           ;+Tabanfang = 1.Buchstabe in Tab.
1E30  73                LD      (HL),E          ;Typcode in Tabelle eintragen
1E31  23                INC     HL              ;Tabellenadresse + 1
1E32  3D                DEC     A               ;Zähler - 1
1E33  20 FB             JR      NZ,1E30H        ;fertig ? nein-nächster Buchstabe
1E35  E1                POP     HL              ;Programmzeiger laden
1E36  7E                LD      A,(HL)          ;Zeichen aus Programmtext laden
1E37  FE 2C             CP      ','             ;folgen weitere Parameter ?
1E39  C0                RET     NZ              ;nein, fertig
1E3A  D7                RST     10H             ;nächstes Zeichen laden
1E3B  18 CE             JR      1E0BH           ;weitere Definitionen eintragen

                        ************************************

                        Testet, ob Zeichen ein Buchstabe ist
                        Eing.: HL = Adresse des zu untersuchenden Zeichens
                        Ausg.: Cy = 0 - Buchstabe, Cy = 1 - kein Buchstabe
1E3D  7E                LD      A,(HL)          ;Zeichen laden
1E3E  FE 41             CP      'A'             ;< A ?
```

```
1E40  D8           RET   C             ;ja, kein Buchstabe
1E41  FE 5B        CP    5BH           ;<= Z  ja, Carry = 1
1E43  3F           CCF                 ;Carry invertieren
1E44  C9           RET

                   ************************************

                   Ausdruck auswerten und ganzzahligen Wert
                   < 32768 ermitteln.
                   Eing.: HL = Textadresse - 1
                   Ausg.: DE = Ergebnis
1E45  D7           RST   10H           ;nächstes Zeichen adressieren
1E46  CD 02 2B     CALL  2B02H         ;Ausdruck auswerten
1E49  F0           RET   P             ;> 32767 ? nein, fertig

                   ************************************

                   FUNCTION CODE - Error
1E4A  1E 08        LD    E,8           ;Fehlercode in E
1E4C  C3 A2 19     JP    19A2H         ;Fehlermeldung ausgeben

                   ************************************

                   String in Zahl umwandeln ( < 65530 )
                   Eing.: HL = Adresse des Strings
                   Ausg.: DE = Zahl
1E4F  7E           LD    A,(HL)        ;Zeichen aus String laden
1E50  FE 2E        CP    2EH           ;= '.' ?
1E52  EB           EX    DE,HL         ;Stringzeiger in DE
1E53  2A EC 78     LD    HL,(78ECH)    ;'.'-Zeilennummer in HL
1E56  EB           EX    DE,HL         ;Zeiger und '.'-ZNr tauschen
1E57  CA 78 1D     JP    Z,1D78H       ;ja, fertig
1E5A  2B           DEC   HL            ;Stringzeiger - 1
1E5B  11 00 00     LD    DE,0          ;Zahl = 0 setzen
1E5E  D7           RST   10H           ;nächstes Zeichen laden
1E5F  D0           RET   NC            ;keine Ziffer, fertig
1E60  E5           PUSH  HL            ;Stringzeiger auf Stack
1E61  F5           PUSH  AF            ;Zeichen auf Stack
1E62  21 98 19     LD    HL,1998H      ;Zahl > 1998H ?
1E65  DF           RST   18H           ;(d.h. Zahl*10 > 65529)
1E66  DA 97 19     JP    C,1997H       ;ja, SYNTAX ERROR
1E69  62           LD    H,D           ;Zahl in HL umladen
1E6A  6B           LD    L,E
1E6B  19           ADD   HL,DE         ;Zahl * 2
```

```
1E6C  29          ADD   HL,HL          ;     * 4
1E6D  19          ADD   HL,DE          ;     * 5
1E6E  29          ADD   HL,HL          ;     * 10
1E6F  F1          POP   AF             ;Zeichen wieder laden
1E70  D6 30       SUB   30H            ;Zonenteil entfernen
1E72  5F          LD    E,A            ;in DE übertragen
1E73  16 00       LD    D,0
1E75  19          ADD   HL,DE          ;auf 10*Zahl addieren
1E76  EB          EX    DE,HL          ;Zahl in DE übertragen
1E77  E1          POP   HL             ;Stringzeiger laden
1E78  18 E4       JR    1E5EH          ;nächste Ziffer

                  ************************************

                  CLEAR - Anweisung
                  Variable löschen und Stringbereich definieren
1E7A  CA 61 1B    JP    Z,1B61H        ;keine Parameter? Sprung in NEW
1E7D  CD 46 1E    CALL  1E46H          ;Ausdruck auswerten
1E80  2B          DEC   HL             ;Programmzeiger - 1
1E81  D7          RST   10H            ;nächstes Zeichen adressieren
1E82  C0          RET   NZ             ;Anweisungsende? nein, Fehler
1E83  E5          PUSH  HL             ;Programmzeiger auf Stack
1E84  2A B1 78    LD    HL,(78B1H)     ;BASIC-RAM Endadresse laden
1E87  7D          LD    A,L            ;- Argument der CLEAR-Anweisung
1E88  93          SUB   E              ;= Start des String-Bereichs - 1
1E89  5F          LD    E,A
1E8A  7C          LD    A,H
1E8B  9A          SBC   A,D
1E8C  57          LD    D,A
1E8D  DA 7A 19    JP    C,197AH        ;Unterlauf, OUT OF MEMORY - Error
1E90  2A F9 78    LD    HL,(78F9H)     ;Startadr. der Variablen-Tabelle
1E93  01 28 00    LD    BC,40          ;+ 64
1E96  09          ADD   HL,BC
1E97  DF          RST   18H            ;< neue Stringbereichs-Adresse - 1?
1E98  D2 7A 19    JP    NC,197AH       ;nein, OUT OF MEMORY - Error
1E9B  EB          EX    DE,HL          ;neue Stringbereichs-Startadr - 1
1E9C  22 A0 78    LD    (78A0H),HL     ;abspeichern
1E9F  E1          POP   HL             ;Programmzeiger laden
1EA0  C3 61 1B    JP    1B61H          ;weiter bei NEW

                  ************************************

                  RUN - Anweisung
                  Programm starten
```

```
1EA3  CA 5D 1B        JP      Z,1B5DH         ;keine Zeilennummer? weiter bei NEW
1EA6  CD C7 79        CALL    79C7H           ;RAM-Erweiterungsausgang
1EA9  CD 61 1B        CALL    1B61H           ;Variable löschen
1EAC  01 1E 1D        LD      BC,1D1EH        ;Rücksprungadresse laden
1EAF  18 10           JR      1EC1H           ;weiter bei GOTO

                      ************************************

                      GOSUB - Anweisung
                      Unterprogramm aufrufen
1EB1  0E 03           LD      C,3             ;testen, ob noch 6 Bytes frei sind
1EB3  CD 63 19        CALL    1963H
1EB6  C1              POP     BC              ;Rücksprungadresse löschen
1EB7  E5              PUSH    HL              ;Programmzeig. für RETURN auf Stack
1EB8  E5              PUSH    HL              ;Programmzeiger nachmals auf Stack
1EB9  2A A2 78        LD      HL,(78A2H)      ;mit aktueller Zeilennummer
1EBC  E3              EX      (SP),HL         ;vertauschen
1EBD  3E 91           LD      A,91H           ;91 als Flag für GOSUB
1EBF  F5              PUSH    AF              ;auf den Stack
1EC0  33              INC     SP              ;LSB entfernen
1EC1  C5              PUSH    BC              ;Rücksprungadr. wieder auf Stack

                      ************************************

                      GOTO - Anweisung
                      unbedingter Sprung
1EC2  CD 5A 1E        CALL    1E5AH           ;Sprung-Zeilennummer ermitteln
1EC5  CD 07 1F        CALL    1F07H           ;Ende der Zeile suchen
1EC8  E5              PUSH    HL              ;Programmzeiger auf Stack
1EC9  2A A2 78        LD      HL,(78A2H)      ;aktuelle Zeilennummer in HL
1ECC  DF              RST     18H             ;= Sprung < Zeilennummer ?
1ECD  E1              POP     HL              ;Programmzeiger laden
1ECE  23              INC     HL              ;auf Anfang der nächsten Zeile
1ECF  DC 2F 1B        CALL    C,1B2FH         ;ja, Sprungzeile von dieser Zeile
                                              ;an suchen
1ED2  D4 2C 1B        CALL    NC,1B2CH        ;nein, Sprungzeile vom Programm-
                                              ;anfang an suchen
1ED5  60              LD      H,B             ;Adresse der Sprungzeile in HL
1ED6  69              LD      L,C
1ED7  2B              DEC     HL              ;Programmzeiger vor Sprungzeile
1ED8  D8              RET     C               ;Zeile vorhanden? ja, dort weiter

                      ************************************
```

```
                        UNDEFINED STATEMENT - Error
1ED9  1E 0E             LD      E,0EH           ;Fehlercode in E
1EDB  C3 A2 19          JP      19A2H           ;Fehlermeldung ausgeben

                        ************************************

                        RETURN - Anweisung
                        Rücksprung von einem Unterprogramm
1EDE  C0                RET     NZ              ;Parameter? ja, Fehler
1EDF  16 FF             LD      D,0FFH          ;Daten vom Stack zurückholen
1EE1  CD 36 19          CALL    1936H           ;(FOR - Daten dabei übergehen)
1EE4  F9                LD      SP,HL           ;Stack neu initialisieren
1EE5  22 E8 78          LD      (78E8H),HL
1EE8  FE 91             CP      91H             ;Daten von einem GOSUB-Aufruf?
1EEA  1E 04             LD      E,4             ;Code für RETURN WITHOUT GOSUB Err.
1EEC  C2 A2 19          JP      NZ,19A2H        ;nein, Fehlermeldung ausgeben
1EEF  E1                POP     HL              ;Zeilennummer vom Stack laden
1EF0  22 A2 78          LD      (78A2H),HL      ;als aktuelle Zeilennummer speich.
1EF3  23                INC     HL              ;Direkt - Mode ?
1EF4  7C                LD      A,H             ;(=FFFF)
1EF5  B5                OR      L
1EF6  20 07             JR      NZ,1EFFH        ;nein!
1EF8  3A DD 78          LD      A,(78DDH)       ;RESUME/RETURN-Flag gesetzt ?
1EFB  B7                OR      A
1EFC  C2 18 1A          JP      NZ,1A18H        ;ja, zurück zur Hauptschleife
1EFF  21 1E 1D          LD      HL,1D1EH        ;Rücksprungadresse laden
1F02  E3                EX      (SP),HL         ;mit Programmzeiger tauschen
1F03  3E                DEFB    3EH             ;LD A,0E1H  Dummy-Befehl

1F04  E1                POP     HL              ;Programmzeiger laden

                        ************************************

                        DATA - Anweisung
                        Ende der Anweisung suchen
1F05  01 3A 0E          LD      BC,0E3AH        ;Trennzeichen1 = ':' in C
1F08  00                NOP

                        ************************************

                        ELSE - Anweisung
                        Zeilenende suchen
1F07  0E 00             LD      C,0             ;Trennzeichen1 = 00 in C
                                                ;Achtung: 1F07-1F08 redefiniert
```

```
1F09  06 00      LD    B,0          ;Trennzeichen2 = 00 in B
1F0B  79         LD    A,C          ;Trennzeichen 1 und 2 tauschen
1F0C  48         LD    C,B
1F0D  47         LD    B,A
1F0E  7E         LD    A,(HL)       ;Zeichen laden
1F0F  B7         OR    A            ;= Zeilenende ?
1F10  C8         RET   Z            ;ja, fertig
1F11  B8         CP    B            ;= Trennzeichen2 ?
1F12  C8         RET   Z            ;ja, fertig
1F13  23         INC   HL           ;Programmzeiger + 1
1F14  FE 22      CP    '"'          ;= Anführungszeichen ?
1F16  28 F3      JR    Z,1F0BH      ;ja, Trennzeichen tauschen
                                    ;(d.h. nur noch Zeilenende suchen)
1F18  D6 8F      SUB   8FH          ;IF - Token?
1F1A  20 F2      JR    NZ,1F0EH     ;nein, weiter
1F1C  B8         CP    B            ;wenn nicht im String oder nach
                                    ;ELSE, Carry = 1 setzen
1F1D  8A         ADC   A,D          ;Verschachtelungszähler + 1
1F1E  57         LD    D,A
1F1F  18 ED      JR    1F0EH        ;weiter

                 ************************************

                 LET - Anweisung
                 Wertzuweisung
1F21  CD 0D 26   CALL  260DH        ;Variable in Tabelle suchen
1F24  CF         RST   8            ;Folgt das Zeichen '=' ?
1F25  D5         DEFB  '='
1F26  EB         EX    DE,HL        ;Adresse der Variablen-Tabelle
1F27  22 DF 78   LD    (78DFH),HL   ;für Variable merken
1F2A  EB         EX    DE,HL
1F2B  D5         PUSH  DE           ;und auf den Stack packen
1F2C  E7         RST   20H          ;Typ testen
1F2D  F5         PUSH  AF           ;Typ-Flag auf Stack
1F2E  CD 37 23   CALL  2337H        ;Ausdruck auswerten
1F31  F1         POP   AF           ;Typ-Flag laden
1F32  E3         EX    (SP),HL      ;Programmzeiger auf Stack
                                    ;Adresse in Variablen-Tabelle laden
1F33  C6 03      ADD   A,3          ;TypCode errechnen
1F35  CD 19 28   CALL  2819H        ;Ergebnis des Ausdrucks in
                                    ;richtigen Typ umwandeln
1F38  CD 03 0A   CALL  0A03H        ;X-Adresse in DE
1F3B  E5         PUSH  HL           ;Adr.der Variablen-Tab. auf Stack
1F3C  20 28      JR    NZ,1F66H     ;Sprung, wenn nicht String
```

```
                        Stringzuweisung
1F3E  2A 21 79          LD      HL,(7921H)      ;Stringzeiger aus X-Reg laden
1F41  E5                PUSH    HL              ;und auf Stack
1F42  23                INC     HL              ;Stringadresse laden
1F43  5E                LD      E,(HL)          ;in DE
1F44  23                INC     HL
1F45  56                LD      D,(HL)
1F46  2A A4 78          LD      HL,(78A4H)      ;String nicht im Programmtext oder
                                                ;im Stringbereich ?
1F49  DF                RST     18H
1F4A  30 0E             JR      NC,1F5AH        ;ja, String in Stringbereich
1F4C  2A A0 78          LD      HL,(78A0H)      ;String im Programmtext?
1F4F  DF                RST     18H
1F50  D1                POP     DE              ;Stringzeiger laden
1F51  30 0F             JR      NC,1F62H        ;ja, String nicht in Stringbereich!
1F53  2A F9 78          LD      HL,(78F9H)      ;zeigt Stringzeiger auf Var.Tab.?
1F56  DF                RST     18H
1F57  30 09             JR      NC,1F62H        ;nein, String nicht im Stringber.
1F59  3E                DEFB    3EH             ;LD A,0D1H  Dummy-Befehl
155A  D1                POP     DE              ;Stringzeiger laden
1F5B  CD F5 29          CALL    29F5H           ;String im Zwischenspeicher löschen
1F5E  EB                EX      DE,HL           ;Stringzeiger in HL
1F5F  CD 43 28          CALL    2843H           ;String in Stringbereich übertragen
1F62  CD F5 29          CALL    29F5H           ;String im Zwischenspeicher löschen
1F65  E3                EX      (SP),HL         ;Zeiger auf Zwischenspeicher auf
                                                ;Stack, Var.Tab.-Adresse laden
1F66  CD D3 09          CALL    09D3H           ;Wert von X in Variablen-Tabelle
1F69  D1                POP     DE              ;Stack  bereinigen
1F6A  E1                POP     HL              ;Programmzeiger laden
1F6B  C9                RET

                        ************************************

                        ON - Anweisung
                        Verzweigung anhand einer Sprungleiste
1F6C  FE 9E             CP      9EH             ;folgt ein ERROR-Token ?
1F6E  20 25             JR      NZ,1F95H        ;nein!

                        ON ERROR
1F70  D7                RST     10H             ;nächstes Zeichen adressieren
1F71  CF                RST     8               ;ist es ein GOTO-Token?
1F72  8D                DEFB    8DH             ;(8D = GOTO-Token)
1F73  CD 5A 1E          CALL    1E5AH           ;Zeilennummer dekodieren
```

```
1F76  7A          LD    A,D          ;= 0?
1F77  B3          OR    E            ;(Fehlerbehandlung ausschalten)
1F78  28 09       JR    Z,1F83H      ;ja!
1F7A  CD 2A 1B    CALL  1B2AH        ;Zeile im Programmtext suchen
1F7D  50          LD    D,B          ;Zeilenadresse in DE
1F7E  59          LD    E,C
1F7F  E1          POP   HL           ;Programmzeiger laden
1F80  D2 D9 1E    JP    NC,1ED9H     ;Zeile nicht vorhanden!
                                     ;UNDEFINED STATEMENT - Error
1F83  EB          EX    DE,HL        ;Adresse der Fehlerroutine
1F84  22 F0 78    LD    (78F0H),HL   ;abspeichern
1F87  EB          EX    DE,HL
1F88  D8          RET   C            ;Zeilen-Nr > 0, fertig!
1F89  3A F2 78    LD    A,(78F2H)    ;schon ein Fehler aufgetreten?
1F8C  B7          OR    A
1F8D  C8          RET   Z            ;nein, fertig
1F8E  3A 9A 78    LD    A,(789AH)    ;Fehlercode in E
1F91  5F          LD    E,A
1F92  C3 AB 19    JP    19ABH        ;zur Fehlerbehandlung

                  ON GOTO - ON GOSUB
1F95  CD 1C 2B    CALL  2B1CH        ;Ausdruck auswerten, Ganzzahliger
                                     ;Wert (< 256) in E
1F98  7E          LD    A,(HL)       ;Zeichen aus Programmtext laden
1F99  47          LD    B,A          ;in B
1F9A  FE 91       CP    91H          ;= GOSUB - Token ?
1F9C  28 03       JR    Z,1FA1H      ;ja!
1F9E  CF          RST   8            ;ist es ein GOTO - Token ?
1F9F  8D          DEFB  8DH          ;(8D = GOTO-Token)
1FA0  2B          DEC   HL           ;Programmzeiger - 1
1FA1  4B          LD    C,E          ;Sprungvariable in C
1FA2  0D          DEC   C            ;Variable - 1 = 0 ?
1FA3  78          LD    A,B          ;Token in A f. Sprungausführung
1FA4  CA 60 1D    JP    Z,1D60H      ;ja, Sprung m. n. Zeilen-Nr ausführ
1FA7  CD 5B 1E    CALL  1E5BH        ;Zeilennummer dekodieren
1FAA  FE 2C       CP    ','          ;folgt ein Komma ?
1FAC  C0          RET   NZ           ;nein, Programm mit der nächsten
                                     ;Anweisung fortsetzen
1FAD  18 F3       JR    1FA2H        ;nächste Zeilennummer
```

****

RESUME - Anweisung
Rücksprung von der Fehlerbehandlung

```
1FAF  11 F2 78     LD    DE,78F2H      ;TRAP-Flag adressieren
1FB2  1A           LD    A,(DE)        ;Fehler aufgetreten ?
1FB3  B7           OR    A
1FB4  CA A0 19     JP    Z,19A0H       ;nein, RESUME WITHOUT ERROR
1FB7  3C           INC   A             ;A = 0
1FB8  32 9A 78     LD    (789AH),A     ;Fehlercode löschen
1FBB  12           LD    (DE),A        ;TRAP-Flag löschen
1FBC  7E           LD    A,(HL)        ;Zeichen laden
1FBD  FE 87        CP    87H           ;= NEXT-Token ?
1FBF  28 0C        JR    Z,1FCDH       ;ja! RESUME NEXT
1FC1  CD 5A 1E     CALL  1E5AH         ;Zeilennummer dekodieren
1FC4  C0           RET   NZ            ;weitere Zeichen? ja-Fehler
1FC5  7A           LD    A,D           ;Zeilennummer = 0 ?
1FC6  B3           OR    E
1FC7  C2 C5 1E     JP    NZ,1EC5H      ;nein, bei GOTO fortsetzen
1FCA  3C           INC   A             ;A = 1
1FCB  18 02        JR    1FCFH

                   RESUME NEXT
1FCD  D7           RST   10H           ;nächstes Zeichen im Programmtext
1FCE  C0           RET   NZ            ;kein Zeilenende, Fehler
1FCF  2A EE 78     LD    HL,(78EEH)    ;Zeiger auf fehlerhafte Zeile
1FD2  EB           EX    DE,HL         ;in DE
1FD3  2A EA 78     LD    HL,(78EAH)    ;Fehler-Zeilennummer laden
1FD6  22 A2 78     LD    (78A2H),HL    ;als aktuelle Zeilennummer eintrag.
1FD9  EB           EX    DE,HL         ;Zeiger wieder in HL
1FDA  C8           RET   NZ            ;RESUME 0? ja-fertig
1FDB  7E           LD    A,(HL)        ;Zeilenende?
1FDC  B7           OR    A
1FDD  20 04        JR    NZ,1FE3H      ;nein, nächste Anweisung in Zeile
1FDF  23           INC   HL            ;Programmzeiger auf 1. Anweisung
1FE0  23           INC   HL            ;der nächsten Zeile
1FE1  23           INC   HL            ;(hinter Zeiger und Zeilennummer)
1FE2  23           INC   HL
1FE3  23           INC   HL
1FE4  7A           LD    A,D           ;Direkt-Mode ?
1FE5  A3           AND   E             ;(Zeilennummer = FFFF)
1FE6  3C           INC   A
1FE7  C2 05 1F     JP    NZ,1F05H      ;nein, nächste Anweisung, fertig
1FEA  3A DD 78     LD    A,(78DDH)     ;RETURN/RESUME-Flag gesetzt?
1FED  3D           DEC   A
1FEE  CA BE 1D     JP    Z,1DBEH       ;ja, Prorammausführung beenden.
1FF1  C3 05 1F     JP    1F05H         ;nächste Anweisung suchen, fertig
```

```
                        ************************************

                        ERROR - Anweisung
                        erzeugt angegebenen Fehler
1FF4  CD 1C 2B          CALL    2B1CH           ;Fehlercode analysieren
                                                ;Wert (<256) in A
1FF7  C0                RET     NZ              ;weitere Zeichen? ja-Fehler
1FF8  B7                OR      A               ;Fehlercode = 0 ?
1FF9  CA 4A 1E          JP      Z,1EAAH         ;ja, FUNCTION CODE - Error
1FFC  3D                DEC     A               ;Internen Fehlercode ermitteln
1FFD  87                ADD     A,A
1FFE  5F                LD      E,A             ;und in E ablegen
1FFF  FE 2D             CP      2DH             ;< 2D ?
2001  38 02             JR      C,2005H         ;ja!

                        ************************************

                        UNPRINTABLE ERROR
2003  1E 26             LD      E,26H           ;Fehlercode in E
2005  C3 A2 19          JP      19A2H           ;zur Fehler-Routine

                        ************************************

                        AUTO - Anweisung
                        Automatische Zeilennummerierung
2008  11 0A 00          LD      DE,10           ;Anfangs- und Erhöhungswert = 10
200B  D5                PUSH    DE              ;auf den Stack
200C  28 17             JR      Z,2025H         ;keine weiteren Zeichen eingegeben!
200E  CD 4F 1E          CALL    1E4FH           ;Anfangswert dekodieren
2011  EB                EX      DE,HL           ;Anfangswert in HL, Prog-zeiger DE
2012  E3                EX      (SP),HL         ;Anfangswert auf den Stack
                                                ;10 als Erhöhungswert in HL
2013  28 11             JR      Z,2026H         ;keine weiteren Zeichen eingegeben!
2015  EB                EX      DE,HL           ;Programmzeiger in HL
2016  CF                RST     8               ;folgt ein Komma ?
2017  2C                DEFB    ','
2018  EB                EX      DE,HL           ;Programmzeiger wieder in DE
2019  2A E4 78          LD      HL,(78E4H)      ;alten Erhöhungswert laden
201C  EB                EX      DE,HL           ;Programmzeiger in HL
201D  28 06             JR      Z,2025H         ;keine weiteren Zeichen nach Komma!
201F  CD 5A 1E          CALL    1E5AH           ;Erhöhungswert dekodieren
2022  C2 97 19          JP      NZ,1997H        ;Zeilenende? nein-SYNTAX ERROR
2025  EB                EX      DE,HL           ;Erhöhungswert in HL
2026  7C                LD      A,H             ;= 0 ?
```

```
2027  B5          OR    L
2028  CA 4A 1E    JP    Z,1E4AH       ;ja, FUNCTION CODE - Error
202B  22 EA 78    LD    (78E4H),HL    ;Erhöhungswert speichern
202E  32 E1 78    LD    (78E1H),A     ;AUTO - Flag setzen
2031  E1          POP   HL            ;Anfangswert laden
2032  2           LD    (78E2H),HL    ;und abspeichern
2035  C1          POP   BC            ;Rücksprungadresse vom Stack holen
2036  C3 33 1A    JP    1A33H         ;zur Hauptschleife

                  ************************************

                  IF - Anweisung
                  Bedingungs-Abfrage
2039  CD 37 23    CALL  2337H         ;Bedingungsausdruck auswerten
203C  7E          LD    A,(HL)        ;Zeichen laden
203D  FE 2C       CP    ','           ;= Komma ?
203F  CC 78 1D    CALL  Z,1D78H       ;ja, nächstes Zeichen
2042  FE CA       CP    0CAH          ;= THEN - Token ?
2044  CC 78 1D    CALL  Z,1D78H       ;ja, nächstes Zeichen
2047  2B          DEC   HL            ;Programmzeiger - 1
2048  E5          PUSH  HL            ;und auf den Stack
2049  CD 94 09    CALL  0994H         ;Ergebnis = 0 ? (nicht erfüllt!)
204C  E1          POP   HL            ;Programmzeiger wieder laden
204D  28 07       JR    Z,2056H       ;ja, zur ELSE - Ausführung

                  THEN
204F  D7          RST   10H           ;nächstes Zeichen
2050  DA C2 1E    JP    C,1EC2H       ;Ziffer? ja-Sprung ausführen
2053  C3 5F 1D    JP    1D5FH         ;nein, nächste Anweisung ausf.

                  ELSE
2056  16 01       LD    D,1           ;Verschachtelungszähler = 1
2058  CD 05 1F    CALL  1F05H         ;nächste Anweisung suchen,
                                      ;wenn IF, Versch.zähler + 1
205B  B7          OR    A             ;Zeilenende?
205C  C8          RET   Z             ;fertig, kein ELSE
205D  D7          RST   10H           ;nächstes Zeichen
205E  FE 95       CP    95H           ;= ELSE - Token?
2060  20 F6       JR    NZ,2058H      ;nein, weiter suchen
2062  15          DEC   D             ;richtiges ELSE ?
                                      ;(Verschachtelungs-Zähler - 1 = 0)
2063  20 F3       JR    NZ,2058H      ;nein, weiter suchen
2065  18 E8       JR    204FH         ;ja, weiter wie THEN
```

```
                        ************************************

                        LPRINT - Anweisung
                        Ausgabe auf dem Drucker
2067  3E 01             LD      A,1             ;Ausgabe-Flag = Drucker
2069  32 9C 78          LD      (789CH),A
206C  C3 9B 20          JP      209BH           ;weiter bei PRINT

                        ************************************

                        PRINT - Anweisung
                        Ausgabe auf dem Bildschirm
206F  CD CA 79          CALL    79CAH           ;RAM-Erweiterungsausgang
2072  FE 40             CP      '@'             ;PRINT @ ?
2074  20 19             JR      NZ,208FH        ;nein!

2076  CD 01 2B          CALL    2B01H           ;Positionsausdruck auswerten
                                                ;Wert (<32768) in DE, MSB in A
2079  FE 02             CP      2               ;Position > 511 ?
207B  D2 4A 1E          JP      NC,1E4AH        ;ja, FUNCTION CODE - Error
207E  E5                PUSH    HL              ;Programmzeiger auf Stack
207F  21 00 70          LD      HL,7000H        ;Bildschirm-Startadresse laden
2082  19                ADD     HL,DE           ;Position dazu addieren
2083  22 20 78          LD      (7820H),HL      ;als neue Cursoradresse speichern
2086  7B                LD      A,E             ;Position des Cursors in Zeile erm.
2087  E6 1F             AND     1FH             ;= 5 letzten Bits d. Cursoradresse
2089  32 A6 78          LD      (78A6H),A       ;als neue Cursorposition speichern
208C  E1                POP     HL              ;Programmzeiger laden
208D  CF                RST     8               ;folgt ein Komma?
208E  2C                DEFB    ','
208F  FE 23             CP      '#'             ;Kassettenausgabe ?
2091  20 08             JR      NZ,209BH        ;nein, weiter
2093  CD 58 3B          CALL    3B58H           ;Vorspann auf Kassette schreiben
2096  3E 80             LD      A,80H           ;Ausgabe-Flag auf Kassette
2098  32 9C 78          LD      (789CH),A
209B  2B                DEC     HL              ;Programmzeiger - 1
209C  D7                RST     10H             ;nächstes Zeichen. Anweisungsende?
209D  CC FE 20          CALL    Z,20FEH         ;ja, CR ausgeben
20A0  CA 69 21          JP      Z,2169H         ;und fertig
20A3  FE BF             CP      0BFH            ;= USING - Token ?
20A5  CA BD 2C          JP      Z,2CBDH         ;ja, formatierte Ausgabe
20A8  FE BC             CP      0BCH            ;= TAB - Token ?
20AA  CA 37 21          JP      Z,2137H         ;ja!
20AD  E5                PUSH    HL              ;Programmzeiger auf Stack
```

```
20AE  FE 2C       CP    ','          ;Komma ?
20B0  CA 08 21    JP    Z,2108H      ;ja, zur nächsten TAB-Position
20B3  FE 3B       CP    ';'          ;Semikolon ?
20B5  CA 0C 3B    JP    Z,3B0CH      ;warten, bis alle Zeichen ausgegeb.
                                     ;Fortsetzung bei 2164H
20B8  C1          POP   BC           ;Programmzeiger laden
20B9  CD 37 23    CALL  2337H        ;Ausdruck auswerten
20BC  E5          PUSH  HL           ;Programmzeiger auf Stack
20BD  E7          RST   20H          ;Datentyp testen
20BE  28 32       JR    Z,20F2H      ;String ? ja, Sprung
20C0  CD BD 0F    CALL  0FBDH        ;num. Werte in String umwandeln
20C3  CD 65 28    CALL  2865H        ;String in Zwischenspeicher und X
20C6  CD CD 79    CALL  79CDH        ;RAM-Erweiterungsausgang
20C9  2A 21 79    LD    HL,(7921H)   ;Stringzeiger aus X laden
20CC  3A 9C 78    LD    A,(789CH)    ;Ausgabe-Flag laden
20CF  B7          OR    A            ;und testen
20D0  FA E9 20    JP    M,20E9H      ;Kassette? ja-keine Formatierung
20D3  28 08       JR    Z,20DDH      ;Bildschirm? ja-Sprung
20D5  3A 9B 78    LD    A,(789BH)    ;Druckkopf-Position laden
20D8  86          ADD   A,(HL)       ;+ Stringlänge
20D9  FE 84       CP    84H          ;> Zeilenlänge (132) ?
20DB  18 09       JR    20E6H        ;weiter bei 20E6H
20DD  3A 9D 78    LD    A,(789DH)    ;Bildschirm-Zeilenlänge laden
                                     ;(wird mit 64 initialisiert.)
20E0  47          LD    B,A          ;in B
20E1  3A A6 78    LD    A,(78A6H)    ;Cursorposition in Zeile laden
20E4  86          ADD   A,(HL)       ;+ Stringlänge
20E5  B8          CP    B            ;> Zeilenlänge (64) ?
20E6  DA FE 20    CALL  NC,20FEH     ;ja, Carriage Return ausgeben
20E9  CD AA 28    CALL  28AAH        ;String ausgeben
20EC  3E 20       LD    A,' '        ;danach ein Leerzeichen
20EE  CD 2A 03    CALL  032AH
20F1  B7          OR    A            ;Z=0, damit wird n. Befehl übersp.
20F2  CC AA 28    CALL  Z,28AAH      ;String ausdrucken
20F5  E1          POP   HL           ;Programmzeiger laden
20F6  C3 9B 20    JP    209BH        ;weiter!

                  Prüfen, ob Cursor am Zeilenanfang steht
20F9  CD 1C 3B    CALL  3B1CH        ;Cursorposition laden
20FC  B7          OR A               ;= 0 ?
20FD  C8          RET   Z            ;ja, zurück

                  Carriage-Return ausgeben
20FE  3E 0D       LD    A,0DH        ;CR-Code laden
```

```
2100  CD 2A 03      CALL    032AH           ;und ausgeben
2103  CD D0 79      CALL    79D0H           ;RAM-Erweiterungsausgang
2106  AF            XOR     A               ;A + Flags rücksetzen
2107  C9            RET

                    ',' auswerten
2108  CD D3 79      CALL    79D3H           ;RAM-Erweiterungsausgang
210B  3A 9C 78      LD      A,(789CH)       ;Ausgabe-Flag laden
210E  B7            OR      A               ;Bildschirm oder Drucker ?
210F  F2 19 21      JP      P,2119H         ;ja!
2112  3E 2C         LD      A,','           ;Komma auf Kassette aufzeichnen
2114  CD 2A 03      CALL    032AH
2117  18 4B         JR      2164H           ;weiter bei 2164H
2119  28 0B         JR      Z,2123H         ;Bildschirm ? ja - Sprung
211B  3A 9B 78      LD      A,(789BH)       ;Kopfposition < letzte Tabposition?
211E  FE 70         CP      70H             ;(= 112)
2120  C3 2B 21      JP      212BH           ;weiter bei 212BH
2123  3A 9E 78      LD      A,(789EH)       ;letzte Tabposition laden (48!)
2126  47            LD      B,A             ;in B
2127  3A AE 7A      LD      A,(7AA6H)       ;Cursorposition laden (ungepuffert)
212A  B8            CP      B               ;< letzte Tabposition ?
212B  D4 FE 20      CALL    NC,20FEH        ;nein, Carriage-Return ausgeben
212E  30 34         JR      NC,2164H        ;und weiter
2130  D6 10         SUB     16              ;Cursorposition - 16 bis < 0
2132  30 FC         JR      NC,2130H
2134  2F            CPL                     ;= Anzahl einzuf. Leerzeichen -1
2135  18 23         JR      215AH           ;Leerzeichen ausgeben

                    TAB auswerten
2137  CD 1B 2B      CALL    2B1BH           ;Ausdruck auswerten
                                            ;ganzz. Wert (<256) in A
213A  E6 7F         AND     7FH             ;Bit 7 löschen (max 127)
213C  5F            LD      E,A             ;in E
213D  CF            RST     8               ;folgt ein ')' ?
213E  29            DEFB    ')'
213F  2B            DEC     HL              ;Programmzeiger - 1
2140  E5            PUSH    HL              ;und auf den Stack
2141  CD D3 79      CALL    79D3H           ;RAM-Erweiterungsausgang
2144  3A 9C 78      LD      A,(789CH)       ;Ausgabe-Flag laden
2147  B7            OR      A               ;und testen
2148  FA 4A 1E      JP      M,1E4AH         ;Kassette? FUNCTION CODE - Error
214B  CA 53 21      JP      Z,2153H         ;Bildschirm? ja-Sprung
214E  3A 9B 78      LD      A,(789BH)       ;Druckkopfposition laden
2151  18 03         JR      2156H           ;weiter bei Bildschirm
```

```
2153  3A A6 78        LD    A,(78A6H)      ;Cursorposition laden
2156  2F              CPL                  ;1er Komplement bilden
2157  83              ADD   A,E            ;+ Tab-Wert
2158  30 0A           JR    NC,2164H       ;bereits erreicht oder überschr.
215A  3C              INC   A              ;+ 1
215B  47              LD    B,A            ;= Anzahl einzufügender Leerzeichen
215C  3E 20           LD    A,' '          ;Leerzeichen ausgeben
215E  CD 2A 03        CALL  032AH
2161  05              DEC   B              ;Zähler - 1
2162  20 FA           JR    NZ,215EH       ;=0? nein - nächstes Leerzeichen

                      Nächsten PRINT - Unterausdruck
2164  E1              POP   HL             ;Programmzeiger laden
2165  D7              RST   10H            ;nächstes Zeichen adressieren
2166  C3 A0 20        JP    20A0H          ;und zurück

                      Endabfrage
2169  3A 9C 78        LD    A,(789CH)      ;Ausgabe-Flag laden
216C  00 00 00 00     DEFB  0,0,0,0        ;4 x NOP
2170  AF              XOR   A              ;Ausgabe-Flag auf
2171  32 9C 78        LD    (789CH),A      ;Bildschirm setzen
2174  CD BE 79        CALL  79BEH          ;RAM-Erweiterungsausgang

                      ************************************

                      Textdefinition
2178  3F 52 45 44 4F  DEFM  '?REDO'
217D  0D 00           DEFW  000DH

                      ************************************

                      Fehler beim Einlesen von Daten
217F  3A DE 78        LD    A,(78DEH)      ;DATA-Flag gesetzt ?
2182  B7              OR    A
2183  C2 91 19        JP    NZ,1991H       ;ja, SYNTAX ERROR in DATA-Anweisung
2186  3A A9 78        LD    A,(78A9H)      ;Eingabe von Kassette ?
2189  B7              OR    A
218A  1E 2A           LD    E,2AH          ;Fehlercode in E
218C  CA A2 19        JP    Z,19A2H        ;ja, BAD FILE DATA - Error
218F  C1              POP   BC             ;Tastatureingabe, Pufferzeiger lad.
2190  21 78 21        LD    HL,2178H       ;Text '?REDO' adressieren
2193  CD A7 28        CALL  28A7H          ;und ausgeben
2196  2A E6 78        LD    HL,(78E6H)     ;Aktuellen Prog.zeiger in HL
2199  C9              RET                  ;Eingabe neu beginnen
```

```
                        ************************************

                        INPUT - Anweisung
                        Daten einlesen
219A  CD 28 28          CALL    2828H           ;Direktbefehl?
                                                ;ja, ILLEGAL DIRECT OPERATION
219D  7E                LD      A,(HL)          ;Zeichen laden
219E  CD D6 79          CALL    79D6H           ;RAM-Erweiterungsausgang
21A1  D6 23             SUB     '#'             ;Lesen von Kassette ?
21A3  32 A9 78          LD      (78A9H),A       ;Differenz als INPUT-Flag (0=Kass)
21A6  7E                LD      A,(HL)          ;Zeichen laden
21A7  20 20             JR      NZ,21C9H        ;keine Kassette!

                        Einlesen von Kassette
21A9  CD 68 3B          CALL    3B68H           ;Datei auf Kassette suchen
21AC  E5                PUSH    HL              ;Programmzeiger auf Stack
21AD  06 FA             LD      B,0FAH          ;max. 250 Zeichen
21AF  2A A7 78          LD      HL,(78A7H)      ;Ein-/Ausgabe-Puffer adressieren
21B2  CD 88 3B          CALL    3B88H           ;ein Byte lesen
21B5  77                LD      (HL),A          ;in Puffer übertragen
21B6  23                INC     HL              ;Pufferzeiger + 1
21B7  FE 0D             CP      0DH             ;Satzende ?
21B9  28 02             JR      Z,21BDH         ;ja!
21BB  10 F5             DJNZ    21B2H           ;Zähler - 1 = 0 ?
21BD  2B                DEC     HL              ;ja,Satzende mit 00 kennzeichnen
21BE  36 00             LD      (HL),0
21C0  00 00 00          DEFB    0,0,0           ;3 * NOP
21C3  2A A7 78          LD      HL,(78A7H)      ;Pufferanfang adressieren
21C6  2B                DEC     HL              ;Pufferzeiger 1 Byte vor Anfang
21C7  18 22             JR      21EBH           ;weiter bei 21EBH

                        Einlesen von der Tastatur
21C9  01 DB 21          LD      BC,21DBH        ;Rücksprungadresse setzen
21CC  C5                PUSH    BC
21CD  FE 22             CP      '"'             ;mit vorheriger Textausgabe ?
21CF  C0                RET     NZ              ;nein, weiter bei 21DBH
21D0  CD 66 28          CALL    2866H           ;Text in Zwischenspeicher u. X
21D3  CF                RST     8               ;folgt ein Semikolon?
21D4  3B                DEFB    ';'
21D5  E5                PUSH    HL              ;Programmzeiger auf den Stack
21D6  CD AA 28          CALL    28AAH           ;Text ausgeben
21D9  E1                POP     HL              ;Programmzeiger laden
21DA  C9                RET                     ;weiter bei 21DBH
```

```
21DB  E5           PUSH  HL          ;Programmzeiger auf Stack
21DC  CD B3 1B     CALL  1BB3H       ;'?' drucken und eine Zeile in
                                     ;den Ein-/Ausgabepuffer einlesen
21DF  C1           POP   BC          ;Programmzeiger in BC
21E0  DA BE 1D     JP    C,1DBEH     ;BREAK? ja - Sprung
21E3  23           INC   HL          ;Pufferzeiger auf 1. Zeichen
21E4  7E           LD    A,(HL)      ;Zeichen laden
21E5  B7           OR    A           ;Textende ?
21E6  2B           DEC   HL          ;Pufferzeiger wieder vor 1. Zeichen
21E7  C5           PUSH  BC          ;Programmzeiger auf Stack
21E8  CA 04 1F     JP    Z,1F04H     ;kein Text, INPUT-Anweis. übergehen
21EB  36 2C        LD    (HL),','    ;Komma vor erstes Zeichen setzen
21ED  18 05        JR    21F4H       ;weiter bei 21F4H

                   READ - Anweisung
                   Daten aus dem Programmtext lesen
21EF  E5           PUSH  HL          ;Programmzeiger auf Stack
21F0  2A FF 78     LD    HL,(78FFH)  ;DATA - Zeiger in HL
21F3  F6 AF        OR    0AFH        ;DATA - Flag setzen

21F4  AF           XOR   A           ;DATA - Flag löschen
                                     ;Achtung: Redefinition von 21F4H
21F5  32 DE 78     LD    (78DEH),A   ;DATA - Flag abspeichern
21F8  E3           EX    (SP),HL     ;Puffer-/DATA-Zeiger auf Stack
                                     ;Programmzeiger laden
21F9  18 02        JR    21FDH       ;weiter bei 21FDH

                   Nächste Variable
21FB  CF           RST   8           ;folgt ein Komma ?
21FC  2C           DEFB  ','
21FD  CD 0D 26     CALL  260DH       ;Variable in Var.Tabelle suchen
                                     ;Var.Tab.Adresse in DE
2200  E3           EX    (SP),HL     ;Programmzeiger auf Stack
                                     ;Pufferzeiger laden
2201  D5           PUSH  DE          ;Var.Tab. Adresse auf Stack
2202  7E           LD    A,(HL)      ;Zeichen aus dem Puffer laden
2203  FE 2C        CP    ','         ;= Komma ?
2205  28 26        JR    Z,222DH     ;ja, weiter

                   Puffer leer (kein ',')
2207  3A DE 78     LD    A,(78DEH)   ;DATA - Flag gesetzt ?
220A  B7           OR    A
220B  C2 96 22     JP    NZ,2296H    ;ja, nächste DATA-Anweisung suchen
220E  3A A9 78     LD    A,(78A9H)   ;Eingabe von Kassette ?
```

```
2211  B7           OR    A
2212  1E 06        LD    E,6          ;Fehlercode in E
2214  CA A2 19     JP    Z,19A2H      ;ja, OUT OF DATA - Error
2217  3E 3F        LD    A,'?'        ;Tastatur: '?' ausgeben
2219  CD 2A 03     CALL  032AH
221C  CD B3 1B     CALL  1BB3H        ;Erneute Eingabe mit '??
221F  D1           POP   DE           ;Var.Tabellen-Adresse laden
2220  C1           POP   BC           ;Programmzeiger in BC
2221  DA BE 1D     JP    C,1DBEH      ;BREAK? ja - Sprung
2224  23           INC   HL           ;Pufferzeiger auf 1. Zeichen
2225  7E           LD    A,(HL)       ;Zeichen laden
2226  B7           OR    A            ;Zeilenende ?
2227  2B           DEC   HL           ;Pufferzeiger vor 1. Zeichen
2228  C5           PUSH  BC           ;Programmzeiger auf Stack
2229  CA 04 1F     JP    Z,1F04H      ;ja, restliche Eingabe übergehen,
                                      ;ohne Variablenwerte zu ändern
222C  D5           PUSH  DE           ;Var.Tab.-Adresse wieder auf Stack

                   Eingabe dekodieren
222D  CD DC 79     CALL  79DCH        ;RAM-Erweiterungsausgang
2230  E7           RST   20H          ;Typ der Variablen testen
2231  F5           PUSH  AF           ;Typ-Flag sichern
2232  20 19        JR    NZ,224DH     ;numerisch? ja, Sprung

                   String übernehmen
2234  D7           RST   10H          ;Pufferzeiger auf nächstes Zeichen
2235  57           LD    D,A          ;als Trennzeichen in D und B
2236  47           LD    B,A
2237  FE 22        CP    '"'          ;Anführungszeichen ?
2239  28 05        JR    Z,2240H      ;ja, '"' als Trennzeichen benutzen
223B  16 3A        LD    D,':'        ;nein ':' und ',' als Trennzeichen
223F  2B           DEC   HL           ;Pufferzeiger 1 Byte zurück
2240  CD 69 28     CALL  2869H        ;String in Zwischenspeicher und X

                   Neuen Variablenwert abspeichern
2243  F1           POP   AF           ;Typ-Flag laden
2244  EB           EX    DE,HL        ;Pufferzeiger in DE
2245  21 5A 22     LD    HL,225AH     ;Rücksprungadresse in HL
2248  E3           EX    (SP),HL      ;mit Var.Tab.-Adr auf Stack tausch.
2249  D5           PUSH  DE           ;Pufferzeiger auf Stack
224A  C3 33 1F     JP    1F33H        ;Sprung in LET und dann 225AH

                   Zahl in ▮ übernehmen
224D  D7           RST   10H          ;Nächstes Zeichen adressieren
```

```
224E  F1              POP   AF          ;Typ-Flag laden
224F  F5              PUSH  AF          ;und wieder auf Stack
2250  01 43 22        LD    BC,2243H    ;Rücksprungadresse auf Stack
2253  C5              PUSH  BC
2254  DA 6C 0E        JP    C,0E6CH     ;Integer und einf.Genauigkeit?
                                        ;ja, String umwandeln, dann 2243H
2257  D2 65 0E        JP    NC,0E65H    ;dopp.Genauigkeit ? umw., dann 2243
225A  2B              DEC   HL          ;Pufferzeiger - 1
225B  D7              RST   10H         ;nächstes Zeichen. 00 oder ':' ?
225C  28 05           JR    Z,2263H     ;ja, Zeilenende!
225E  FE 2C           CP    ','         ;Komma ?
2260  C2 7F 21        JP    NZ,217FH    ;nein, Fehler
2263  E3              EX    (SP),HL     ;Programmzeiger m. Pufferzeiger
                                        ;auf dem Stack tauschen
2264  2B              DEC   HL          ;Programmzeiger - 1
2265  D7              RST   10H         ;nächstes Zeichen. = Anw.ende?
2266  C2 FB 21        JP    NZ,21FBH    ;nein, weiter m. n. Variablen

                      keine weiteren Variablen
2269  D1              POP   DE          ;Pufferzeiger in DE
226A  00 00 00 00 00  DEFB  0,0,0,0,0   ;5 x NOP
226F  3A DE 78        LD    A,(78DEH)   ;DATA-Flag laden
2272  B7              OR    A           ;gesetzt ?
2273  EB              EX    DE,HL       ;Pufferzeiger-HL,Progr.zeiger-DE
2274  C2 96 1D        JP    NZ,1D96H    ;Pufferzeiger als DATA-Zeiger sp.
                                        ;Programmzeiger in HL, fertig
2277  D5              PUSH  DE          ;Programmzeiger auf Stack
2278  CD DF 79        CALL  79DFH       ;RAM-Erweiterungsausgang
227B  B6              OR    (HL)        ;Zeilenende im Puffer ?
227C  21 86 22        LD    HL,2286H    ;Text '?EXTRA IGNORED' adressieren
227F  C4 A7 28        CALL  NZ,28A7H    ;nein, Text ausgeben
2282  E1              POP   HL          ;Programmzeiger laden
2283  C3 69 21        JP    2169H       ;Ausgabe-Flag auf Bildschirm,fertig

2286  3F 45 58 54     DEFM  '?EXTRA IGNORED'
      52 41 20 49
      47 4E 4F 52
      45 44
2294  0D 00           DEFW  000DH

                      Nächste DATA-Anweisung suchen
2296  CD 05 1F        CALL  1F05H       ;Ende der Anweisung suchen
2299  B7              OR    A           ;= Zeilenende ?
229A  20 12           JR    NZ,22AEH    ;nein!
```

```
229C  23           INC   HL            ;ja, Programmende ?
229D  7E           LD    A,(HL)        ;(Zeilenzeiger = 0000)
229E  23           INC   HL
229F  B6           OR    (HL)
22A0  1E 06        LD    E,6           ;Fehlercode in E
22A2  CA A2 19     JP    Z,19A2H       ;ja, OUT OF DATA - Error
22A5  23           INC   HL            ;Zeilennummer laden
22A6  5E           LD    E,(HL)
22A7  23           INC   HL
22A8  56           LD    D,(HL)
22A9  EB           EX    DE,HL         ;in HL
22AA  22 DA 78     LD    (78DAH),HL    ;und als DATA-ZNr abspeichern
22AD  EB           EX    DE,HL         ;Zeilenummer wieder in DE
22AE  D7           RST   10H           ;nächstes Zeichen aus Programmtext
22AF  FE 88        CP    88H           ;DATA - Token ?
22B1  20 E3        JR    NZ,2296H      ;nein, weiter suchen
22B3  C3 2D 22     JP    222D          ;Daten weiter lesen

                   ************************************

                   NEXT - Anweisung
                   Wiederholung bei FOR-NEXT - Schleifen
22B6  11 00 00     LD    DE,0          ;Var.Tab.-Adresse = 0 setzen
                                       ;(für NEXT ohne Variable)
22B9  C4 0D 26     CALL  NZ,260DH      ;weitere Zeichen ? ja - Variable
                                       ;suchen, Var.Tab.-Adresse in DE
22BC  22 DF 78     LD    (78DFH),HL    ;Programmzeiger speichern
22BF  CD 36 19     CALL  1936H         ;im Stack nächste, oder Schleife
                                       ;mit richtiger Laufvariablen suchen
22C2  C2 9D 1)     JP    NZ,199DH      ;nicht gefunden, NEXT WITHOUT FOR
22C5  F9           LD    SP,HL         ;durch Stackkorrektur alle dazw.
22C6  22 E8 78     LD    (78E8H),HL    ;verschachtelten Schleifen entfern.
22C9  D5           PUSH  DE            ;Var.Tab.-Adr der Laufvar. auf St.
22CA  7E           LD    A,(HL)        ;Erhöhungs-Flag laden
22CB  23           INC   HL            ;Stackzeiger + 1
22CC  F5           PUSH  AF            ;Erhöhungs-Flag auf Stack
22CD  D5           PUSH  DE            ;Var.Tab.-Adresse auf Stack
22CE  7E           LD    A,(HL)        ;Typ-Flag laden
22CF  23           INC   HL            ;Stackzeiger + 1
22D0  B7           OR    A             ;= einfache Genauigkeit?
22D1  FA EA 22     JP    M,22EAH       ;nein! - Sprung

                   Laufvariable einfacher Genauigkeit
22D4  CD B1 09     CALL  09B1H         ;Erhöhungswert in
```

```
22D7  E3          EX    (SP),HL     ;Var.Tab.Adresse laden
                                    ;Stackzeiger auf den Stack
22D8  E5          PUSH  HL          ;Var.Tab. Adresse wieder auf Stack
22D9  CD 0B 07    CALL  070BH       ;Laufvariable+Erhöhungswert
22DC  E1          POP   HL          ;Var.Tab.-Adresse laden
22DD  CD CB 09    CALL  09CBH       ;neuen Wert d. Laufvar. speichern
22E0  E1          POP   HL          ;Stackzeiger laden
22E1  CD C2 09    CALL  09C2H       ;Endwert in Y laden
22E4  E5          PUSH  HL          ;Stackzeiger auf den Stack
22E5  CD 0C 0A    CALL  0A0CH       ;Laufvariable mit Endwert vergl.
22E8  18 29       JR    2313H       ;weiter bei 2313H

                  Integer als Laufvariable
22EA  23          INC   HL          ;2 unben. Stackebenen übergehen
22EB  23          INC   HL
22EC  23          INC   HL
22ED  23          INC   HL
22EE  4E          LD    C,(HL)      ;Erhöhungswert in BC
22EF  23          INC   HL          ;Stackzeiger + 1
22F0  46          LD    B,(HL)      ;(MSB)
22F1  23          INC   HL          ;Stackzeiger + 1
22F2  E3          EX    (SP),HL     ;Stackzeiger auf den Stack
                                    ;Var.Tab.-Adr. der Laufveriabl. lad
22F3  5E          LD    E,(HL)      ;Wert der Laufvariablen laden
22F4  23          INC   HL
22F5  56          LD    D,(HL)
22F6  E5          PUSH  HL          ;Var.Tab.-Adr + 1 auf den Stack
22F7  69          LD    L,C         ;Erhöhungswert in HL
22F8  60          LD    H,B
22F9  CD D2 0B    CALL  0BD2H       ;Laufvar.+Erhöhungswert in HL u. X
22FC  3A AF 78    LD    A,(78AFH)   ;Typ in X = einf.Genauigk.?
22FF  FE 04       CP    4           ;(überlauf)
2301  CA B2 07    JP    Z,07B2H     ;ja, OVERFLOW - Error
2304  EB          EX    DE,HL       ;neue Laufvariable in DE
2305  E1          POP   HL          ;Var.Tab.-Adr + 1 laden
2306  72          LD    (HL),D      ;und neuen Wert eintragen
2307  2B          DEC   HL
2308  73          LD    (HL),E
2309  E1          POP   HL          ;Stackzeiger laden
230A  D5          PUSH  DE          ;neuen Wert der Laufvar. auf Stack
230B  5E          LD    E,(HL)      ;Endwert laden
230C  23          INC   HL
230D  56          LD    D,(HL)
230E  23          INC   HL          ;Stackzeiger + 1
```

```
230F  E3           EX    (SP),HL      ;Stackzeiger auf den Stack
                                      ;neuen Wert der Laufvariablen laden
2310  CD 39 0A     CALL  0A39H        ;Laufvariable mit Endwert vergleich

2313  E1           POP   HL           ;Stackzeiger laden
2314  C1           POP   BC           ;Erhöhungs-Flag laden
2315  90           SUB   B            ;Vergleichsergebnis mit Erhöhungs-
                                      ;Flag verknüpfen
2316  CD C2 09     CALL  09C2H        ;Zeilennummer und Anfangszeiger
                                      ;in DE und BC laden
2319  28 09        JR    Z,2324H      ;Schleife beendet ? ja - Sprung
231B  EB           EX    DE,HL        ;Zeilennummer in HL
231C  22 A2 78     LD    (78A2H),HL   ;als aktuelle ZNr. speichern
231F  69           LD    L,C          ;Anfangszeiger der Schleife in HL
2320  60           LD    H,B
2321  C3 1A 1D     JP    1D1AH        ;Schleife erneut durchlaufen

                   Schleife beendet
2324  F9           LD    SP,HL        ;durch Stackkorrektur Schleife
2325  22 E8 78     LD    (78E8H),HL   ;vom Stack entfernen
2328  2A DF 78     LD    HL,(78DFH)   ;Programmzeiger laden
232B  7E           LD    A,(HL)       ;Zeichen laden
232C  FE 2C        CP    ','          ;folgt ein Komma ?
232E  C2 1E 1D     JP    1D1EH        ;nein, nächsten Befehl
2331  D7           RST   10H          ;nächstes Zeichen adressieren
2332  CD B9 22     CALL  22B9H        ;nächste äußere Schleife bearbeiten

                   ************************************

                   Ausdruck auswerten
                   Eing.: HL = Anfangsadresse im Programmtext
                   Ausg.: X = Ergebnis
2335  CF           RST   8            ;beginnt der Ausdruck mit einer
2336  28           DEFB  '('          ;Klammer? nein - SYNTAX ERROR

2337  2B           DEC   HL           ;Programmzeiger - 1
2338  16 00        LD    D,0          ;Prio-Code d. letzten Operanden = 0
233A  D5           PUSH  DE           ;Prio-Code auf den Stack
233B  0E 01        LD    C,1          ;mindestens noch 2 Byte frei?
233D  CD 63 19     CALL  1963H        ;nein, OUT OF MEMORY - Error
2340  CD 9F 24     CALL  249FH        ;Operanden analysieren und in X
2343  22 F3 78     LD    (78F3H),HL   ;Programmzeiger speichern
2346  2A F3 78     LD    HL,(78F3H)   ;Programmzeiger laden
2349  C1           POP   BC           ;Prio.-Code in B laden
```

```
234A  7E           LD    A,(HL)      ;nächstes Zeichen aus Programm
234B  16 00        LD    D,0         ;Operator-Code = 0 setzen
234D  D6 D4        SUB   D4H         ;Vergleichsoperator ? ( > = < )
234F  38 13        JR    C,2364H     ;nein!
2351  FE 03        CP    3           (Token D4, D5 und D6)
2353  30 0F        JR    NC,2364H    ;nein!
2355  FE 01        CP    1           ;für '>' Carry setzen
2357  17           RLA               ;1 Bit links schieben
                                     ;( > - 1 , = - 2 , < - 4)
2358  AA           XOR   D           ;entspr. Bit im Operator-Code setz.
2359  BA           CP    D           ;war vorher schon gesetzt ?
235A  57           LD    D,A         ;(d.h. der gleiche Operator 2x)
235B  DA 97 19     JP    C,1997H     ;ja - SYNTAX ERROR
235E  22 D8 78     LD    (78D8H),HL  ;Programmzeiger speichern
2361  D7           RST   10H         ;nächstes Zeichen laden
2362  18 E9        JR    234DH       ;und untersuchen

2364  7A           LD    A,D         ;Operator-Code > 0 ?
2365  B7           OR    A           ;(Vergleichsoperator gefunden)
2366  C2 EC 23     JP    NZ,23ECH    ;ja!
2369  7E           LD    A,(HL)      ;Zeichen laden
236A  22 D8 78     LD    (78D8H),HL  ;Programmzeiger speichern
236D  D6 CD        SUB   0CDH        ;einer der anderen Operatoren ?
236F  D8           RET   C           ;+ - * / ** AND OR ?
2370  FE 07        CP    7
2372  D0           RET   NC          ;nein!
2373  5F           LD    E,A         ;Operator-Code in E
2374  3A AF 78     LD    A,(78AFH)   ;Ist in X ein String ?
2377  D6 03        SUB   3
2379  B3           OR    E           ;und '+' - Operator?
237A  CA 8F 29     JP    Z,298FH     ;ja, Stringverknüpfung
237D  21 9A 18     LD    HL,189AH    ;Tabelle der Prio.-Codes adress.
2380  19           ADD   HL,DE       ;+ Operatorcode
2381  78           LD    A,B         ;letzte Priorität in A
2382  56           LD    D,(HL)      ;neue Prio. aus Tabelle in D
2383  BA           CP    D           ;letzte Prio. >= neue Prio. ?
2384  D0           RET   NC          ;ja, letzte Operation ausführen
2385  C5           PUSH  BC          ;nein, letzte Prio. auf Stack
2386  01 46 23     LD    BC,2346H    ;Adr. f. n. Operanden auf Stack
2389  C5           PUSH  BC
238A  7A           LD    A,D         ;Neue Priorität in A
238B  FE 7F        CP    7FH         ;= 7FH (Operator = **) ?
238D  CA D4 23     JP    Z,23D4H     ;ja, weiter bei 23D4H
2390  FE 51        CP    51H         ;Operator = AND oder OR?
```

```
2392  DA E1 23        JP      C,23E1H         ;ja, weiter bei 23E1H

                      Operanden für +, -, * und / auf den Stack
2395  21 21 79        LD      HL,7921H        ;X-Adresse in HL
2398  B7              OR      A               ;Carry löschen
2399  3A AF 78        LD      A,(78AFH)       ;Typ-Code laden
239C  3D              DEC     A               ;Typ-Code - 3
239D  3D              DEC     A               ;= String ?
239E  3D              DEC     A
239F  CA F6 0A        JP      Z,0AF6H         ;ja, TYPE MISMATCH - Error
23A2  4E              LD      C,(HL)          ;Operanden laden
23A3  23              INC     HL              ;X-Adresse + 1
23A4  46              LD      B,(HL)          ;n. Byte
23A5  C5              PUSH    BC              ;und auf Stack
23A6  FA C5 23        JP      M,23C5H         ;bei Integer fertig!
23A9  23              INC     HL              ;sonst 2 weitere Bytes laden
23AA  4E              LD      C,(HL)
23AB  23              INC     HL
23AC  46              LD      B,(HL)
23AD  C5              PUSH    BC              ;und auf den Stack
23AE  F5              PUSH    AF              ;Typ-Flag auf den Stack
23AF  B7              OR      A               ;= einfache Genauigkeit?
23B0  E2 C4 23        JP      PO,23C4H        ;ja, fertig
23B3  F1              POP     AF              ;Typ-Flag wieder laden
23B4  23              INC     HL              ;X-Adresse + 1
23B5  38 03           JR      C,23BAH         ;Y auf Stack ? ja - Sprung
23B7  21 1D 79        LD      HL,791DH        ;LSB X-Adresse laden
23BA  4E              LD      C,(HL)          ;2 weitere Bytes laden
23BB  23              INC     HL
23BC  46              LD      B,(HL)
23BD  23              INC     HL
23BE  C5              PUSH    BC              ;2 Bytes auf den Stack
23BF  4E              LD      C,(HL)          ;und noch 2 Bytes laden
23C0  23              INC     HL
23C1  46              LD      B,(HL)
23C2  C5              PUSH    BC              ;auch auf den Stack
23C3  06              DEFB    06H             ;LD B,0F1H  Dummy-Befehl
                                              ;überspringt den POP
23C4  F1              POP     AF              ;Typ-Flag laden (bei einf.Gen.)
23C5  C6 03           ADD     A,3             ;Typ-Code berechnen
23C7  4B              LD      C,E             ;Operator-Code in C
23C8  47              LD      B,A             ;Typ-Code in B
23C9  C5              PUSH    BC              ;auf den Stack packen
23CA  01 06 24        LD      BC,2406H        ;Adresse zur Durckführung der
```

```
23CD  C5           PUSH  BC            ;Operationen auf den Stack
23CE  2A D8 78     LD    HL,(78D8H)    ;Pogrammzeiger laden
23D1  C3 3A 23     JP    233AH         ;nächster Operand

                   Operanden für Potenzieren auf den Stack
23D4  CD B1 0A     CALL  0AB1H         ;X in einfache Genauigkeit umwand.
23D7  CD A4 09     CALL  09A4H         ;X auf den Stack
23DA  01 F2 13     LD    BC,13F2H      ;Adresse zur Potenzberechnung
23DD  16 7F        LD    D,7FH         ;neuer Prio.-Code = 7F
23DF  18 EC        JR    23CDH

                   Operanden für AND und OR auf den Stack
23E1  D5           PUSH  DE            ;Operator-Code auf Stack
23E2  CD 7F 0A     CALL  0A7FH         ;Operand in Integer umwandeln
23E5  D1           POP   DE            ;Operator-Code laden
23E6  E5           PUSH  HL            ;Operand auf den Stack
23E7  01 E9 25     LD    BC,25E9H      ;Adresse für 'AND'- und 'OR'-Verarb
23EA  18 E1        JR    23CDH

                   Vergleichsoperatoren verarbeiten
23EC  78           LD    A,B           ;hatte letzter Operator höhere
23ED  FE 64        CP    64H           ;oder gleiche Priorität ?
23EF  D0           RET   NC            ;ja, letzte Operation berechnen
23F0  C5           PUSH  BC            ;letzte Priorität auf Stack
23F1  D5           PUSH  DE            ;Operator-Code auf Stack
23F2  11 04 64     LD    DE,6404H      ;Priorität in D,Vergleichscode in E
23F5  21 B8 25     LD    HL,25B8H      ;Adresse zur Bearbeitung des
23F8  E5           PUSH  HL            ;Vergleichsergebnisses auf Stack
23F9  E7           RST   20H           ;Datentyp testen
23FA  C2 95 23     JP    NZ,2395H      ;numerisch? ja-Operanden auf Stack
23FD  2A 21 79     LD    HL,(7921H)    ;String! Stringzeiger auf Stack
2400  E5           PUSH  HL
2401  01 8C 25     LD    BC,258CH      ;Adresse Stringvergleich laden
2404  18 C7        JR    23CDH         ;und auf den Stack

                   Operationen ausführen
2406  C1           POP   BC            ;Operator-Code und Typ laden
2407  79           LD    A,C           ;Operatorcode speichern
2408  32 B0 78     LD    (78B0H),A
240B  78           LD    A,B           ;Typ in A
240C  FE 08        CP    8             ;1. Operand dopp.Gen. ?
240E  28 28        JR    Z,2438H       ;ja!
2410  3A AF 78     LD    A,(78AFH)     ;Typ des 2. Operanden laden
2413  FE 08        CP    8             ;= doppelte Genauigkeit ?
```

```
2415  CA 60 24        JP      Z,2460H         ;ja!
2418  57              LD      D,A             ;Typ des 2. Operanden in D
2419  78              LD      A,B             ;Typ 1. Operand in A
241A  FE 04           CP      4               ;= einfache Genauigkeit ?
241C  CA 72 24        JP      Z,2472H         ;ja!
241F  7A              LD      A,D             ;Typ 2. Operand laden
2420  FE 03           CP      3               ;und testen
2422  CA F6 0A        JP      Z,0AF6H         ;String! - TYPE MISMATCH Error
2425  D2 7C 24        JP      NC,247CH        ;einf. Genauigkeit! - Sprung

                      Integer - Operationen ausführen
2428  21 BF 18        LD      HL,18BFH        ;Start der Sprungtabelle 'Integer'
242B  06 00           LD      B,0             ;Operator-Code 2x addieren
242D  09              ADD     HL,BC
242E  09              ADD     HL,BC
242F  4E              LD      C,(HL)          ;Sprungadresse laden
2430  23              INC     HL
2431  46              LD      B,(HL)
2432  D1              POP     DE              ;1. Operand vom Stack holen
2433  2A 21 79        LD      HL,(7921H)      ;2. Operand aus X laden
2436  C5              PUSH    BC              ;Sprungadresse auf den Stack
2437  C9              RET                     ;und Routine anspringen

                      1. Operand doppelte Genauigkeit
2438  CD DB 0A        CALL    0ADBH           ;2. Operand in dopp.Gen. umwandeln
243B  CD FC 09        CALL    09FCH           ;und in Y übertragen
243E  E1              POP     HL              ;1. Operand vom Stack in X
243F  22 1F 79        LD      (791FH),HL      ;zuerst die 4 niederw. Bytes
2442  E1              POP     HL
2443  22 1D 79        LD      (791DH),HL
2446  C1              POP     BC              ;die n. 3 Bytes und der Exponent
2447  D1              POP     DE
2448  CD B4 09        CALL    09B4H           ;auch in X
244B  CD DB 0A        CALL    0ADBH           ;1. Operand in dopp.Gen. umwandeln
244E  21 AB 18        LD      HL,18ABH        ;Start der Sprungtabelle laden
2451  3A 00 78        LD      A,(7800H)       ;Operator-Code laden
2454  07              RLCA                    ;* 2
2455  C5              PUSH    BC              ;BC sichernk
2456  4F              LD      C,A             ;Operator-Code * 2 in BC
2457  06 00           LD      B,0
2459  09              ADD     HL,BC           ;auf Startadr. der Sprungtab. add.
245A  C1              POP     BC              ;BC wiederherstellen
245B  7E              LD      A,(HL)          ;Sprungadresse laden
245C  23              INC     HL
```

```
245D  66           LD    H,(HL)
245E  6F           LD    L,A
245F  E9           JP    (HL)          ;Ausführungs-Routine anspringen

                   2. Operand = doppelte Genauigkeit
2460  C5           PUSH  BC            ;Operatorcode und Typ auf Stack
2461  CD FC 09     CALL  09FCH         ;2. Operand in Y übertragen
2464  F1           POP   AF            ;Typ des 1. Operanden
2465  32 AF 78     LD    (78AFH),A     ;in Typ-Byte
2468  FE 04        CP    4             ;= einfache Genauigkeit?
246A  28 DA        JR    Z,2446H       ;ja, weiter bei 2446H
246C  E1           POP   HL            ;nein, Integer in HL
246D  22 21 79     LD    (7921H),HL    ;und in X
2470  18 D9        JR    244BH         ;weiter bei 244BH

                   1. Operand = einfache Genauigkeit
2472  CD B1 0A     CALL  0AB1H         ;2. Operand in einf.Gen.umwandeln
2475  C1           POP   BC            ;1. Operand vom Stack in Y
2476  D1           POP   DE
2477  21 B5 18     LD    HL,18B5H      ;Startadresse der Sprungtabelle
247A  18 D5        JR    2451H         ;weiter bei 2451H

                   2. Operand = einfache Genauigkeit
247C  E1           POP   HL            ;1. Operand (Integer) in HL
247D  CD A4 09     CALL  09A4H         ;2. Operand auf Stack
2480  CD CF 0A     CALL  0ACFH         ;1. Operand in einf.Gen.umwandeln
2483  CD BF 09     CALL  09BFH         ;und in Y übertragen
2486  E1           POP   HL            ;2. Operand aus Stack in X
2487  22 23 79     LD    (7923H),HL    ;(Exp. + MSB)
248A  E1           POP   HL
248B  22 21 79     LD    (7921H),HL    ;(2 Byte LSB)
248E  18 E7        JR    2477H         ;weiter bei 2477H

                   ************************************

                   Integer - Division
                   Eing.: DE = Dividend
                          HL = Divisor
                   Ausg.: X  = Quotient (in einfacher Genauigkeit)
2490  E5           PUSH  HL            ;Divisor auf Stack
2491  EB           EX    DE,HL         ;Dividend in HL
2492  CD CF 0A     CALL  0ACFH         ;und mit einf.Gen. in X
2495  E1           POP   HL            ;Divisor laden
2496  CD A4 09     CALL  09A4H         ;Dividend aus X auf Stack
```

```
2499  CD CF 0A        CALL   0ACFH          ;Divisor mit einf.Gen. in X
249C  C3 A0 08        JP     08A0H          ;zur Division mit einf. Genauigk.

                      ************************************

                      Operanden für Ausdrucksanalyse auswerten
                      Eing.: HL = Adresse des Operanden im Text
                      Ausg.: X = Ergebnis
249F  D7              RST    10H            ;nächstes Zeichen adressieren
24A0  1E 28           LD     E,28H          ;Fehlercode in E
24A2  CA A2 19        JP     Z,19A2H        ;Anweis.ende, MISSING OPERAND Error
24A5  DA 6C 0E        JP     C,0E6CH        ;Ziffer! Wert ermitteln und in X
24A8  CD 3D 1E        CALL   1E3DH          ;Buchstabe?
24AB  D2 40 25        JP     NC,2540H       ;ja, Variablenwert in Ausdruck
24AE  FE CD           CP     0CDH           ;'+' - Vorzeichen ?
24B0  28 ED           JR     Z,249FH        ;ja, ignorieren
24B2  FE 2E           CP     '.'            ;Zeichen = '.' ?
24B4  CA 6C 0E        JP     Z,0E6CH        ;ja, Zahl nach X, fertig
24B7  FE CE           CP     0CEH           ;'-' - Vorzeichen ?
24B9  CA 32 25        JP     Z,2532H        ;ja, auswerten
24BC  FE 22           CP     '"'            ;Anführungszeichen ?
24BE  CA 66 28        JP     Z,2866H        ;ja, Stringkonstante in X
24C1  FE CB           CP     0CBH           ;NOT - Token ?
24C3  CA C4 25        JP     Z,24C4H        ;ja, ausführen
24C6  FE 26           CP     '&'            ;= '&' ?
24C8  CA 94 79        JP     Z,7994H        ;ja, zum RAM-Erweiterungsausgang
24CB  FE C3           CP     0C3H           ;= ERR-Token ?
24CD  20 0A           JR     NZ,24D9H       ;nein, weiter

                      ERR - Funktion
                      ergibt den letzten Fehler-Code
24CF  D7              RST    10H            ;nächstes Zeichen adressieren
24D0  3A 9A 78        LD     A,(789AH)      ;letzten Fehlercode laden
24D3  E5              PUSH   HL             ;Programmzeiger auf Stack
24D4  CD F8 27        CALL   27F8H          ;Fehlercode als Integer in X
24D7  E1              POP    HL             ;Programmzeiger wieder laden
24D8  C9              RET                   ;fertig

24D9  FE C2           CP     0C2H           ;= ERL-Token ?
24DB  20 0A           JR     NZ,24E7H       ;nein, weiter

                      ERL - Funktion
                      ermittelt die letzte Fehlerzeile
24DD  D7              RST    10H            ;nächstes Zeichen adressieren
```

```
24DE  E5          PUSH  HL          ;Programmzeiger auf Stack
24DF  2A EA 78    LD    HL,(78EAH)  ;letzte Fehler-Zeilennummer laden
24E2  CD 66 0C    CALL  0C66H       ;in einf.Gen. umwandeln und in X
24E5  E1          POP   HL          ;Programmzeiger wieder laden
24E6  C9          RET               ;fertig

24E7  FE C0       CP    0C0H        ;= VARPTR-Token ?
24E9  20 14       JR    NZ,24FFH    ;nein, weiter

                  VARPTR - Funktion
                  Ermittel Variablen-Adresse in der Variablen-Tabelle
24EB  D7          RST   10H         ;nächstes Zeichen adressieren
24EC  CF          RST   8           ;folgt ein '(' ?
24ED  28          DEFB  '('
24EE  CD 0D 26    CALL  260DH       ;Adresse der Variablen ermitteln
24F1  CF          RST   8           ;abgeschlossen mit ')' ?
24F2  29          DEFB  ')'
24F3  E5          PUSH  HL          ;Programmzeiger auf Stack
24F4  EB          EX    DE,HL       ;Var.Tab.-Adresse in HL
24F5  7C          LD    A,H         ; = 0 ?
24F6  B5          OR    L           ;(Variable nicht in Tabelle)
24F7  CA 4A 1E    JP    Z,1E4AH     ;ja, FUNCTION CODE - Error
24FA  CD 9A 0A    CALL  0A9AH       ;Var.Tab.Adresse als Integer in X
24FD  E1          POP   HL          ;Programmzeiger wieder laden
24FE  C9          RET               ;fertig

24FF  FE C1       CP    0C1H        ;= USR-Token ?
2501  CA FE 27    JP    Z,27FEH     ;ja!
2504  FE C5       CP    0C5H        ;= INSTR$-Token ?
2506  CA 9D 79    JP    Z,799DH     ;ja, zum RAM-Erweiterungsausgang
2509  FE C8       CP    0C8H        ;= MEM-Token ?
250B  CA C9 27    JP    Z,27C9H     ;ja!
250E  FE C7       CP    0C7H        ;= TIME$-Token ?
2510  CA 76 79    JP    Z,7976H     ;ja, zum RAM-Erweiterungsausgang
2513  FE C6       CP    0C6H        ;= POINT-Token ?
2515  CA 32 01    JP    Z,0132H     ;ja!
2518  FE C9       CP    0C9H        ;= INKEY$-Token ?
251A  CA 9D 01    JP    Z,019DH     ;ja!
251D  FE C4       CP    0C4H        ;= STRING$-Token ?
251F  CA 2F 2A    JP    Z,2A2FH     ;ja!
2522  FE BE       CP    0BEH        ;= FN-Token ?
2524  CA 55 79    JP    Z,7955H     ;ja, zum RAM-Erweiterungsausgang
2527  D6 D7       SUB   0D7H        ;Funktions-Token ?
2529  D2 4E 25    JP    NC,254EH    ;ja!
```

```
                        Klammerausdruck auswerten
252C  CD 35 23          CALL    2335H           ;nein, in Klammern eingeschlossenen
                                                ;Ausdruck auswerten
252F  CF                RST     8               ;abgeschlossen mit ')' ?
2530  29                DEFB    ')'
2531  C9                RET                     ;fertig

                        '-' - Vorzeichen auswerten
2532  16 7D             LD      D,7DH           ;Priorität für '-' - Vorzeichen
2534  CD 3A 23          CALL    233AH           ;Ausdruck mit Priorität auswerten
2537  2A F3 78          LD      HL,(78F3H)      ;Programmzeiger laden
253A  E5                PUSH    HL              ;und auf Stack
253B  CD 7B 09          CALL    097BH           ;Ergebnis * (-1)
253E  E1                POP     HL              ;Programmzeiger wieder laden
253F  C9                RET                     ;fertig

                        Variablenwert in Ausdruck einbringen
2540  CD 0D 26          CALL    260DH           ;Variable in Variablen-Tabelle erm.
                                                ;Var.Tab.-Adr. in DE. Wenn nicht
                                                ;vorhanden, X = 0, sofort zurück
2543  E5                PUSH    HL              ;Programmzeiger auf Stack
2544  EB                EX      DE,HL           ;Var.Tab.-Adresse in HL
2545  22 21 79          LD      (7921H),HL      ;und in X als Strngzeiger
2548  E7                RST     20H             ;Stringvariable ?
2549  C4 F7 09          CALL    NZ,09F7H        ;nein, Variablenwert in X
254C  E1                POP     HL              ;Programmzeiger laden
254D  C9                RET                     ;fertig

                        Funktions-Argumente auswerten und
                        Funktions-Routinen anspringen
254E  06 00             LD      B,0             ;B = 0
2550  07                RLCA                    ;A = (Token-D7H) * 2
2551  4F                LD      C,A             ;in BC als Tabellen-Offset
2552  C5                PUSH    BC              ;auf den Stack sichern
2553  D7                RST     10H             ;nächstes Zeichen adressieren
2554  79                LD      A,C             ;LSB Tab-Offset > 41H ?
2555  FE 41             CP      41H             ;(MID$, RIGHT$ o. LEFT$)
2557  38 16             JR      C,256FH         ;nein!
2559  CD 35 23          CALL    2335H           ;1. Argument auswerten
255C  CF                RST     8               ;folgt ein Komma ?
255D  2C                DEFB    ','
255E  CD F4 0A          CALL    0AF4H           ;1. Argument kein String? TM-Error
2561  EB                EX      DE,HL           ;Programmzeiger in DE
```

```
2562  2A 21 79        LD    HL,(7921H)    ;Stringzeiger laden
2565  E3              EX    (SP),HL       ;Tab.Offset laden, Stringz.auf Stk.
2566  E5              PUSH  HL            ;Tab.Offset auch wieder auf Stack
2567  EB              EX    DE,HL         ;Programmzeiger wieder in HL
2568  CD 1C 2B        CALL  2B1CH         ;2. Argument analysieren
                                          ;Ganzz.Wert (<256) in DE
256B  EB              EX    DE,HL         ;2. Arg. in HL, Prog.zeiger in DE
256C  E3              EX    (SP),HL       ;Tab.Offset laden, 2.Arg.auf Stack
256D  18 14           JR    2583H         ;weiter bei 2583H

256F  CD 2C 25        CALL  252CH         ;Argument auswerten
2572  E3              EX    (SP),HL       ;Tab.Offset laden, Prog.z.auf Stack
2573  7D              LD    A,L           ;LSB Tab.Offset < 0CH ?
2574  FE 0C           CP    0CH           ;(SGN, INT, ABS, FRE, INP, POS)
2576  38 07           JR    C,257FH       ;ja!
2578  FE 1B           CP    1BH           ;LSB Tab.Offset < 1BH ?
                                          ;(SQR,RND,LOG,EXP,COS,SIN,TAN,ATN)
257A  E5              PUSH  HL            ;Tab.Offset auf Stack
257B  DC B1 0A        CALL  C,0AB1H       ;ja, Argument in einf.Genauigkeit
257E  E1              POP   HL            ;Tab.Offset wieder laden
257F  11 3E 25        LD    DE,253EH      ;Rücksprungadresse setzen
2582  D5              PUSH  DE
2583  01 08 16        LD    BC,1608H      ;Anfangsadresse der Sprungtabelle
2586  09              ADD   HL,BC         ;+ Tabellen-Offset
2587  4E              LD    C,(HL)        ;Sprungadresse laden
2588  23              INC   HL
2589  66              LD    H,(HL)
258A  69              LD    L,C
258B  E9              JP    (HL)          ;Funktions-Routine anspringen

                      ************************************

                      Stringvergleich
258C  CD D7 29        CALL  29D7H         ;2. String aus Zwischenspeicher
                                          ;und Stringbereich entfernen
258F  7E              LD    A,(HL)        ;Länge 2. String in A
2590  23              INC   HL            ;Adresse 2. String in BC
2591  4E              LD    C,(HL)
2592  23              INC   HL
2593  46              LD    B,(HL)
2594  D1              POP   DE            ;Adresse 1. String in DE
2595  C5              PUSH  BC            ;Adresse 2. String auf Stack
2596  F5              PUSH  AF            ;Länge 2. String auf Stack
2597  CD DE 29        CALL  29DEH         ;1. String aus Zwischenspeicher
```

```
                                        ;und Stringbereich entfernen
259A  D1          POP   DE             ;Länge 2. String nach D
259B  5E          LD    E,(HL)         ;Länge 1. String nach E
259C  23          INC   HL
259D  4E          LD    C,(HL)         ;Adresse 1. String in BC
259E  23          INC   HL
259F  46          LD    B,(HL)
25A0  E1          POP   HL             ;Adresse 2. String in HL
25A1  7B          LD    A,E            ;beide Strings leer?
25A2  B2          OR    D
25A3  C8          RET   Z              ;ja, zurück mit A=0 (gleich)
25A4  7A          LD    A,D            ;Länge 2. String = 0 ?
25A5  D6 01       SUB   1
25A7  D8          RET   C              ;ja, zurück mit A=FF, Cy=1, S=1
                                       ;d.h. String 1 > String 2
25A8  AF          XOR   A              ;2. String = leer ?
25A9  BB          CP    E
25AA  3C          INC   A
25AB  D0          RET   NC             ;ja, zurück mit A=1, Z=0, Cy=0
                                       ;d.h. String 1 < String 2
25AC  15          DEC   D              ;beide Stringlängen - 1
25AD  1D          DEC   E
25AE  0A          LD    A,(BC)         ;Zeichen aus 1. String laden
25AF  BE          CP    (HL)           ;mit Zeichen aus 2. String vergl.
25B0  23          INC   HL             ;Stringzeiger + 1
25B1  03          INC   BC
25B2  28 ED       JR    Z,25A1H        ;beide Zeichen gleich, weiter
25B4  3F          CCF                  ;Carry komplementieren
25B5  C3 60 09    JP    0960H          ;Flag aufbereiten, fertig

                  ************************************

                  Ergebnis des Vergleichs mit dem Vergleichsoperator
                  zum Gesamtergebnis zusammenfassen
                  Eing.: 1. Op. > 2. Op. - A=FF, Cy=1
                         1. Op. = 2. Op. - A=0   1. Op. < 2. Op. - A=1
25B8  3C          INC   A              ;Vergleichsergebnis + 1
25B9  8F          ADC   A,A            ;*2 + Carry
25BA  C1          POP   BC             ;Vergleichs-Operatorcode laden
                                       ;B(0)=1 - >, B(1)=1 - =, B(2)=1 - <
25BB  A0          AND   B              ;stimmt ein Bit überein?
25BC  C6 FF       ADD   A,0FFH         ;ja, Carry = 1
25BE  9F          SBC   A,A            ;und A=FF
25BF  CD 8D 09    CALL  098DH          ;A als Integer in X
```

```
25C2  18 12           JR     25D6H          ;weiter bei 25D6H

                      ************************************

                      NOT ausführen
25C4  16 5A           LD     D,5AH          ;Not-Priorität in D
25C6  CD 3A 23        CALL   233AH          ;Ausdruck mit Priorität auswerten
25C9  CD 7F 0A        CALL   0A7FH          ;Ergebnis in Integer umwandeln
25CC  7D              LD     A,L            ;und invertieren
25CD  2F              CPL                   ;(LSB)
25CE  6F              LD     L,A
25CF  7C              LD     A,H
25D0  2F              CPL                   ;(MSB)
25D1  67              LD     H,A
25D2  22 21 79        LD     (7921H),HL     ;in X übertragen
25D5  C1              POP    BC             ;letzte Priorität laden
25D6  C3 46 23        JP     2346H          ;weiter bei 2346H

                      ************************************

                      Restart 20
                      testet den Datentyp des X-Registers
                      Eing.: 78AF = Typ-Code
                      Ausg.: A = Typ-Code -3
                             Integer: S=1, P=1, Cy=1
                             einf.G.: Cy=1
                             dopp.G.: P=1
                             String : Z=1, C=1, P=1
25D9  3A AF 78        LD     A,(78AFH)      ;Typ-Code laden
25DC  FE 08           CP     8              ;doppelte Genauigkeit ?
25DE  30 05           JR     NC,25E5H       ;ja!
25E0  D6 03           SUB    3              ;Typ-Code - 3
25E2  B7              OR     A              ;Flags setzen
25E3  37              SCF                   ;u.a. Carry=1
25E4  C9              RET
25E5  D6 03           SUB    3              ;Typ-Code - 3
25E7  B7              OR     A              ;Flags setzen
25E8  C9              RET

                      ************************************

                      AND und OR ausführen
25E9  C5              PUSH   BC             ;letzte Priorität auf Stack
25EA  CD 7F 0A        CALL   0A7FH          ;2. Operand in Integer umwandeln
```

```
25ED  F1           POP   AF          ;Priorität in AF laden
25EE  D1           POP   DE          ;1. Operand laden
25EF  01 FA 27     LD    BC,27FAH    ;Rücksprungadresse setzen
25F2  C5           PUSH  BC
25F3  FE 46        CP    46H         ;AND ?
25F5  20 06        JR    NZ,25FDH    ;ja!
25F7  7B           LD    A,E         ;beide Operanden ODER-verknüpfen
25F8  B5           OR    L           ;(LSB)
25F9  6F           LD    L,A
25FA  7C           LD    A,H
25FB  B2           OR    D           ;(MSB)
25FC  C9           RET
25FD  7B           LD    A,E         ;beide Operanden UND-verknüpfen
25FE  A5           AND   L           ;(LSB)
25FF  6F           LD    L,A
2600  7C           LD    A,H
2601  A2           AND   D           ;(MSB)
2602  C9           RET

                   ************************************

                   Weitere Argumente für DIM bereitstellen
2603  2B           DEC   HL          ;Programmzeiger - 1
2604  D7           RST   10H         ;nächstes Zeichen adressieren
2605  C8           RET   Z           ;Anweisungsende? ja-zurück
2606  CF           RST   8           ;folgt ein Komma ?
2607  2C           DEFB  ','

                   ************************************

                   DIM - Anweisung
                   Matrizen einrichten
2608  01 03 26     LD    BC,2603H    ;Rücksprungadresse f. n. Argument
2608  C5           PUSH  BC
260C  F6 AF        OR    AF          ;DIM-Flag setzen

                   ************************************

                   Variable in Tabelle suchen und einrichten, wenn nicht
                   vorhanden
                   Eing.: HL = Adresse Variablenname
                   Ausg.: DE = Adresse in Variablen-Tabelle
260D  AF           XOR   A           ;DIM-Flag löschen
                                     ;Achtung: 260D redefiniert
```

```
260E  32 AE 78        LD    (78AEH),A     ;DIM-Flag speichern

                      Namen ermitteln
2611  46              LD    B,(HL)        ;1.Zeichen des Var.Names in B
2612  CD 3D 1E        CALL  1E3DH         ;Buchstabe ?
2615  DA 97 19        JP    C,1997H       ;nein, SYNTAX-ERROR
2618  AF              XOR   A             ;C (2. Zeichen) löschen
2619  4F              LD    C,A
261A  D7              RST   10H           ;nächstes Zeichen laden
261B  38 05           JR    C,2622H       ;Ziffer ? ja-Sprung
261D  CD 3D 1E        CALL  1E3DH         ;Buchstabe ?
2620  38 09           JR    C,262BH       ;nein, Name nur 1 Buchstabe
2622  4F              LD    C,A           ;2. Zeichen in C
2623  D7              RST   10H           ;nächstes Zeichen laden
2624  38 FD           JR    C,2623H       ;Ziffer? ja-übergehen
2626  CD 3D 1E        CALL  1E3DH         ;Buchstabe?
2629  30 F8           JR    NC,2623H      ;ja, übergehen

                      Typ ermitteln
262B  11 52 26        LD    DE,2652H      ;Rücksprungadresse setzen
262E  D5              PUSH  DE
262F  16 02           LD    D,2           ;Typcode = Integer
2631  FE 25           CD    '%'           ;n. Zeichen = '%' ?
2633  C8              RET   Z             ;ja, fertig
2634  14              INC   D             ;Typcode = String
2635  FE 24           CP    '$'           ;n. Zeichen = '$' ?
2637  C8              RET   Z             ;ja, fertig
2638  00 00 00 00 00  DEFB  0,0,0,0,0,0,0,0,0    ; 9 x NOP
      00 00 00 00
                      Typcode aus Tabelle entnehmen
2641  78              LD    A,B           ;Stellung des 1. Buchstaben im
2642  D6 41           SUB   'A'           ;Alphabet ermitteln
2644  E6 7F           AND   7FH           ;Bit 7 löschen
2646  5F              LD    E,A           ;als Tabellen-Offset in DE
2647  16 00           LD    D,0
2649  E5              PUSH  HL            ;Programmzeiger löschen
264A  21 01 79        LD    HL,7901H      ;Anfang der Typcode-Tabelle adress.
264D  19              ADD   HL,DE         ;+ Offset
264E  56              LD    D,(HL)        ;Typcode aus Tabelle laden
264F  E1              POP   HL            ;Programmzeiger wieder laden
2650  2B              DEC   HL            ;-1, da keine explizite Typangabe
2651  C9              RET                 ;weiter bei 2652H
```

```
2652  7A             LD    A,D          ;Typcode in Typ-Byte übertragen
2653  32 AF 78       LD    (78AFH),A

                     Variable in Variablen-Tabelle suchen
2656  D7             RST   10H          ;nächstes Zeichen adressieren
2657  3A DC 78       LD    A,(78DCH)    ;Indizierung gesperrt ?
265A  B7             OR    A            ;(für Laufvariable)
265B  C2 64 26       JP    NZ,2664H     ;ja!
265E  7E             LD    A,(HL)       ;Zeichen laden
265F  D6 28          SUB   '('          ;= '(' ?
2661  CA E9 26       JP    Z,26E9H      ;ja, indizierte Variable
2664  AF             XOR   A            ;Indizierungs-Sperre aufheben
2665  32 DC 78       LD    (78DCH),A
2668  E5             PUSH  HL           ;Programmzeiger auf Stack
2669  D5             PUSH  DE           ;Typcode auf Stack
266A  2A F9 78       LD    HL,(78F9H)   ;Anfang der Variablen-Tabelle laden
266D  EB             EX    DE,HL
266E  2A FB 78       LD    HL,(78FBH)   ;Ende der Variablen-Tabelle laden
2671  DF             RST   18H          ;Adressen gleich?
2672  E1             POP   HL           ;Typcode in H
2673  28 19          JR    Z,268EH      ;ja, Variable nicht gefunden
2675  1A             LD    A,(DE)       ;Typ aus Variablen-Tabelle laden
2676  6F             LD    L,A          ;in L
2677  BC             CP    H            ;=Typ gesuchter Variabler ?
2678  13             INC   DE           ;Adresse Var.Tabelle + 1
2679  20 0B          JR    NZ,2686H     ;nein, nächste Variable
267B  1A             LD    A,(DE)       ;2. Zeichen aus Tabelle laden
267C  B9             CP    C            ;= 2. Zeichen der Variablen?
267D  20 07          JR    NZ,2686H     ;nein, nächste Variable
267F  13             INC   DE           ;Adresse Var.Tabelle + 1
2680  1A             LD    A,(DE)       ;1. Zeichen aus Tabelle laden
2681  B8             CP    B            ;= 1. Zeichen der Variablen?
2682  CA CC 26       JP    Z,26CCH      ;ja, Variable gefunden!
2685  3E             DEFB  13H          ;LD A,13H  Dummy-Befehl
2686  13             INC   DE           ;Adresse Var.Tabelle auf 1. Zeichen
2687  13             INC   DE           ;Adresse Var.Tabelle auf Wert
2688  E5             PUSH  HL           ;Typ gesuchter Variabler auf Stack
2689  26 00          LD    H,0          ;Adresse der Var.Tabelle
268B  19             ADD   HL,DE        ;+ Länge des Typs = nächster Eintr.
268C  18 DF          JR    266DH        ;weitersuchen

                     Variable nicht in Variablen-Tabelle enthalten
268E  7C             LD    A,H          ;Typ in A
268F  E1             POP   HL           ;Programmzeiger laden
```

```
2690  E3          EX    (SP),HL       ;mit Rücksprungadresse tauschen
2691  F5          PUSH  AF            ;Typ auf Stack
2692  D5          PUSH  DE            ;Endadresse der Var.Tab. auf Stack
2693  11 F1 24    LD    DE,24F1H      ;Rücksprungadresse = 24F1H?
2696  DF          RST   18H           ;(von VARPTR)
2697  28 36       JR    Z,26CFH       ;ja, weiter bei 26CFH
2699  11 43 25    LD    DE,2543H      ;Rücksprungadresse = 2543H?
269C  DF          RST   18H           ;(von Ausdrucksanalyse)
269D  D1          POP   DE            ;Var.Tab.-Endadresse wieder laden
269E  28 35       JR    Z,26D5H       ;ja, weiter bei 26D5H

                  Neue Variable einrichten
26A0  F1          POP   AF            ;Typ laden
26A1  E3          EX    (SP),HL       ;Rücksprungadresse auf den Stack
                                      ;Programmzeiger laden
26A2  E5          PUSH  HL            ;Programmzeiger auf den Stack
26A3  C5          PUSH  BC            ;Variablen-Name auf den Stack
26A4  4F          LD    C,A           ;Typ in C
26A5  06 00       LD    B,0           ;B=0, d.h. BC enth. Länge des Werts
26A7  C5          PUSH  BC            ;Länge auf den Stack
26A8  03          INC   BC            ;+ 3
26A9  03          INC   BC            ;= Gesamtlänge des Var.Tab-Eintrags
26AA  03          INC   BC
26AB  2A FD 78    LD    HL,(78FDH)    ;Anfangsadr. des freien Speichers
26AE  E5          PUSH  HL            ;auf den Stack
26AF  09          ADD   HL,BC         ;+ Gesamtlänge Var.Tab-Eintrag
26B0  C1          POP   BC            ;Wertlänge laden
26B1  E5          PUSH  HL            ;neue Anf.Adr.fr.Speicher auf Stack
26B2  CD 55 19    CALL  1955H         ;Matrix-Tabelle verschieben, um
                                      ;Platz für die neue Variable zu sch
26B5  E1          POP   HL            ;Anf.Adr. freier Speicher laden
26B6  22 FD 78    LD    (78FDH),HL    ;und speichern
26B9  60          LD    H,B           ;neue Anf.Adr. Matrix-Tab. in HL
26BA  69          LD    L,C
26BB  22 FB 78    LD    (78FBH),HL    ;und speichern
26BE  2B          DEC   HL            ;neuen Var.Tab-Eintrag löschen
26BF  36 00       LD    (HL),0        ;(DE = Var.Tab-Adr der Variablen)
26C1  DF          RST   18H           ;fertig ?
26C2  20 FA       JR    NZ,26BEH      ;nein, nächstes Byte
26C4  D1          POP   DE            ;Typ in E laden
26C5  73          LD    (HL),E        ;in Variablen-Tabelle eintragen
26C6  23          INC   HL            ;Tabellenadresse + 1
26C7  D1          POP   DE            ;Name vom Stack holen
26C8  73          LD    (HL),E        ;2. Zeichen in Var.Tabelle
```

```
26C9  23              INC   HL            ;Tabellenadresse + 1
26CA  72              LD    (HL),D        ;1. Zeichen in Var.Tabelle
26CB  EB              EX    DE,HL         ;Tabellenadresse in DE
26CC  13              INC   DE            ;+ 1 = 1. Adresse für Werteintrag
26CD  E1              POP   HL            ;Programmzeiger laden
26CE  C9              RET                 ;fertig

                      bei VARPTR Variable nicht in Tabelle
26CF  57              LD    D,A           ;Var.Tab-Adresse in DE = 0
26D0  5F              LD    E,A
26D1  F1              POP   AF            ;Stack korrigieren
26D2  F1              POP   AF
26D3  E3              EX    (SP),HL       ;Rücksprungadresse auf den Stack
                                          ;Programmzeiger laden
26D4  C9              RET                 ;zurück in VARPTR-Routine

                      bei Ausdrucksanalyse Variable nicht in Tabelle
26D5  32 24 79        LD    (7924H),A     ;X = 0 für einf.u. dopp. Genauigk.
26D8  C1              POP   BC            ;Stack korrigieren
26D9  67              LD    H,A           ;HL = 0 für Integer
26DA  6F              LD    L,A
26DB  22 21 79        LD    (7921H),HL    ;und auch in X eintragen
26DE  E7              RST   20H           ;Typ ermitteln
26DF  20 06           JR    NZ,26E7H      ;String? nein-Sprung
26E1  21 28 19        LD    HL,1928H      ;Stringzeiger auf Leerstring
26E4  22 21 79        LD    (7921H),HL    ;in X eintragen
26E7  E1              POP   HL            ;Programmzeiger laden
26E8  C9              RET                 ;zurück zur Ausdrucksanalyse
                                          ;(2 Stufen)

                      ************************************

                      Matrix - Verwaltung
26E9  E5              PUSH  HL            ;Programmzeiger auf Stack
26EA  2A AE 78        LD    HL,(78AEH)    ;DIM-Flag und Typ laden
26ED  E3              EX    (SP),HL       ;mit Prog.zeiger tauschen
26EE  57              LD    D,A           ;DIM-Zähler = 0
26EF  D5              PUSH  DE            ;DIM-Zähler auf Stack
26F0  C5              PUSH  BC            ;Variablen-Name auf Stack
26F1  CD 45 1E        CALL  1E45H         ;Indizierung auswerten.
                                          ;Ergebnis (<32768) in DE
26F4  C1              POP   BC            ;Variablen-Name laden
26F5  F1              POP   AF            ;DIM-Zähler in A
26F6  EB              EX    DE,HL         ;Index-Wert in HL
```

```
26F7  E3          EX    (SP),HL     ;mit DIM-Flag auf Stack tauschen
26F8  E5          PUSH  HL          ;DIM-Flag und Typ auf Stack
26F9  EB          EX    DE,HL       ;Programmzeiger in HL
26FA  3C          INC   A           ;DIM-Zähler + 1
26FB  57          LD    D,A         ;und in D
26FC  7E          LD    A,(HL)      ;Zeichen laden
26FD  FE 2C       CP    ','         ;folgt ein Komma ?
26FF  28 EE       JR    Z,26EFH     ;ja, nächsten Indexwert

2701  CF          RST   8           ;folgt ein ')' ?
2702  29          DEFB  ')'
2703  22 F3 78    LD    (78F3H),HL  ;Programmzeiger speichern
2706  E1          POP   HL          ;DIM-Flag und Typ laden
2707  22 AE 78    LD    (78AEH),HL  ;und speichern
270A  D5          PUSH  DE          ;DIM-Zähler auf Stack
270B  2A FB 78    LD    HL,(78FBH)  ;Anfang d. Matrixtabelle adress.
270E  3E          DEFB  3EH         ;LD A,19H  Dummy-Befehl
270F  19          ADD   HL,DE       ;Matrixlänge auf Tab.zeiger addier.
2710  EB          EX    DE,HL       ;Adresse der Matrixtab. in HL
2711  2A FD 78    LD    HL,(78FDH)  ;Anfadr. des freien Speichers laden
2714  EB          EX    DE,HL       ;Adressen tauschen
2715  DF          RST   18H         ;Adressen gleich?
2716  3A AF 78    LD    A,(78AFH)   ;Typ laden
2719  28 27       JR    Z,2742H     ;ja, Matrix nicht gefunden!
271B  BE          CP    (HL)        ;Typ = mit Tabelleneintrag?
271C  23          INC   HL          ;Tabellenadresse + 1
271D  20 06       JR    NZ,2727H    ;nein, nächsten Tabelleneintrag
271F  7E          LD    A,(HL)      ;2. Zeichen des Namens aus Tabelle
2720  B9          C     C           ;= 2. Zeichen der gesuchten Matrix?
2721  23          INC   HL          ;Tabellenadresse + 1
2722  20 04       JR    NZ,2728H    ;nein, nächsten Tabelleneintrag
2724  7E          LD    A,(HL)      ;1. Zeichen des Namens aus Tabelle
2725  B8          CP    B           ;= 1. Zeichen der gesuchten Matrix?
2726  3E          DEFB  3EH         ;LD A,23H  Dummy-Befehl
2727  23          INC   HL          ;Tabellenadresse + 1
2728  23          INC   HL          ;Tabellenadresse + 1
2729  5E          LD    E,(HL)      ;Matrixlänge laden
272A  23          INC   HL          ;Tabellenadresse + 1
272B  56          LD    D,(HL)
272C  23          INC   HL          ;Tabellenadresse + 1
272D  20 E0       JR    NZ,270FH    ;1.Zeichen ungleich! n. Tabeintrag

                  Matrix gefunden
272F  3A AE 78    LD    A,(78AEH)   ;DIM-Flag gesetzt ?
```

```
2732  B7           OR    A
2733  1E 12        LD    E,12H          ;Fehlercode in E
2735  C2 A2 19     JP    NZ,19A2H       ;ja, REDIMENSIONED ARRAY - Error
2738  F1           POP   AF             ;DIM-Zähler laden
2739  96           SUB   (HL)           ;=Anzahl Dimensionen in gef.Matrix?
273A  CA 95 27     JP    Z,2795H        ;ja, weiter bei 2795H
273D  1E 10        LD    E,10           ;nein, SUBSCRIPT OUT OF RANGE - Err
273F  C3 A2 19     JP    19A2H          ;zur Fehlerausgabe-Routine

                   Neue Matrix einrichten
2742  77           LD    (HL),A         ;Typ speichern
2743  23           INC   HL             ;Tabellenadresse + 1
2744  5F           LD    E,A            ;Länge eines Elements (=Typ) in DE
2745  16 00        LD    D,0
2747  F1           POP   AF             ;DIM-Zähler laden
2748  71           LD    (HL),C         ;2. Zeichen Matrixname in Tabelle
2749  23           INC   HL             ;Tabellenadresse + 1
274A  70           LD    (HL),B         ;1. Zeichen Matrixname in Tabelle
274B  23           INC   HL             ;Tabellenzeiger + 1
274C  4F           LD    C,A            ;DIM-Zähler in C
274D  CD 63 19     CALL  1963H          ;noch 2*DIM-Zähler Bytes frei?
                                        ;nein, OUT OF MEMORY - Error
2750  23           INC   HL             ;Tabellenadresse + 2
2751  23           INC   HL             ;hinter Längeneintrag
2752  22 D8 78     LD    (78D8H),HL     ;Tabellenadresse speichern
2755  71           LD    (HL),C         ;DIM-Zähler in Tabelle
2756  23           INC   HL             ;Tabellenadresse + 1
2757  3A AE 78     LD    A,(78AEH)      ;DIM-Flag in Carry schieben
275A  17           RLA
275B  79           LD    A,C            ;DIM-Zähler in A
275C  01 0B 00     LD    BC,11          ;Dimension=11 (Standard)
275F  30 02        JR    NC,2763H       ;kein DIM, Sprung
2761  C1           POP   BC             ;Dimension aus Stack in BC
2762  03           INC   BC             ;+ 1 für 0-Index
2763  71           LD    (HL),C         ;in Tabelle eintragen
2764  23           INC   HL             ;Tabellenadresse + 1
2765  70           LD    (HL),B
2766  23           INC   HL             ;Tabellenadresse + 1
2767  F5           PUSH  AF             ;DIM-Zähler auf Stack
2768  CD AA 0B     CALL  0BAAH          ;letzte Matr.-Wertlänge * Dimension
                                        ;zu Beginn = Länge eines Wertes
276B  F1           POP   AF             ;DIM-Zähler laden
276C  3D           DEC   A              ;- 1
276D  20 ED        JR    NZ,275CH       ;weitere Dimensionen? ja-Sprung
```

```
276F  F5           PUSH  AF          ;DIM-Flag (Carry) auf Stack
2770  42           LD    B,D         ;Matrix-Wertlänge in BC
2771  4B           LD    C,E
2772  EB           EX    DE,HL       ;auf Tabellenadresse
                                     ;(jetzt auf 1. Werte-Byte)
2773  19           ADD   HL,DE       ;addieren
2774  38 C7        JR    C,273DH     ;Überlauf, SUBSCRIPT OUT OF RANGE
2776  CD 6C 19     CALL  196CH       ;Ausrreichend Speicher frei ?
                                     ;nein, OUT OF MEMORY - Error
2779  22 FD 78     LD    (78FDH),HL  ;neue Anfadresse 'freier Speicher'
277C  2B           DEC   HL          ;Matrixwerte löschen
277D  36 00        LD    (HL),0      ;durch Einschreiben von 00
277F  DF           RST   18H         ;Werte-Anfangsadresse erreicht?
2780  20 FA        JR    NZ,277CH    ;nein, nächstes Byte
2782  03           INC   BC          ;Matrix-Wertlänge +1 (f. DIM-Zähl.)
2783  57           LD    D,A         ;D = 0
2784  2A D8 78     LD    HL,(78D8H)  ;Adresse des DIM-Zählers laden
2787  5E           LD    E,(HL)      ;DIM-Zahler in E
2788  EB           EX    DE,HL       ;und in HL
2789  29           ADD   HL,HL       ;DIM-Zähler * 2
278A  09           ADD   HL,BC       ;+ Matrix-Wertlänge
278B  EB           EX    DE,HL       ;= Matrixlänge (in DE)
278C  2B           DEC   HL          ;Tabellenadresse - 2
278D  2B           DEC   HL          ;= Zeiger auf Längenfeld
278E  73           LD    (HL),E      ;Matrixlänge in Tabelle eintragen
278F  23           INC   HL          ;Tabellenadresse + 1
2790  72           LD    (HL),D      ;(MSB)
2791  23           INC   HL          ;Tabellenadresse + 1
2792  F1           POP   AF          ;DIM-Flag laden
2793  38 30        JR    C,27C5H     ;DIM? ja-fertig

                   Adresse eines Matrix-Elements ermitteln
2795  47           LD    B,A         ;Matrix-Offset = 0 setzen
2796  4F           LD    C,A         ;in BC
2797  7E           LD    A,(HL)      ;Anzahl Dimensionen in A
2798  23           INC   HL          ;Tabellenzeiger auf 1. Dimension
2799  16           DEFB  16H         ;LD D,0E1H  Dummy-Befehl
279A  E1           POP   HL          ;Tabellenadresse laden
279B  5E           LD    E,(HL)      ;Dimension laden
279C  23           INC   HL          ;Tabellenzeiger + 1
279D  56           LD    D,(HL)      ;(MSB)
279E  23           INC   HL          ;Tabellenzeiger + 1
279F  E3           EX    (SP),HL     ;Tabellenzeiger auf den Stack
                                     ;Index laden
```

```
27A0  F5           PUSH  AF          ;DIM-Zähler auf den Stack
27A1  DF           RST   18H         ;Index >= Dimension ?
27A2  D2 3D 27     JP    NC,273DH    ;ja, SUBSCRIPT OUT OF RANGE - Error
27A5  CD AA 0B     CALL  0BAAH       ;Matrix-Offset * Dimension
27A8  19           ADD   HL,DE       ;+ Index = neues Matrix-Offset
27AA  3D           DEC   A           ;alle Dimensionen verarbeitet?
27AB  44           LD    B,H         ;neuen Matrix-Offset in BC
27AC  4D           LD    C,L
27AD  20 EB        JR    NZ,279AH    ;noch weitere Dimensionen, zurück
27AF  3A AF 78     LD    A,(78AFH)   ;Typ laden (=Wertlänge)
27B2  44           LD    B,H         ;neuen Matrix-Offset in BC
27B3  4D           LD    C,L
27B4  29           ADD   HL,HL       ;Matrix-Offset * 2
27B5  D6 04        SUB   4           ;String oder Integer?
27B7  38 04        JR    C,27BDH     ;ja!
27B9  29           ADD   HL,HL       ;Matrix-Offset * 4
27BA  28 06        JR    Z,27C2H     ;einf.Genauigkeit? ja-Sprung
27BC  29           ADD   HL,HL       ;Matrix-Offset * 8 (dopp.Gen.)
27BD  B7           OR    A           ;Integer oder dopp.Genauigkeit?
27BE  E2 C2 27     JP    PO,27C2H    ;ja!
27C1  09           ADD   HL,BC       ;String, 3 * Matrix-Offset in HL
27C2  C1           POP   BC          ;Werte-Anfangsadresse laden
27C3  09           ADD   HL,BC       ;Matrix-Offset addieren
27C4  EB           EX    DE,HL       ;=Element-Adresse, in DE übertr.
27C5  2A F3 78     LD    HL,(78F3H)  ;Programmzeiger laden
27C8  C9           RET               ;fertig

                   ************************************

                   MEM - Funktion
                   Größe des freien Speichers ermitteln
27C9  AF           XOR   A           ;Typ-Byte = 0 (kein String!)
27CA  E5           PUSH  HL          ;Programmzeiger auf Stack
27CB  32 AF 78     LD    (78AFH),A
27CE  CD D4 27     CALL  27D4H       ;FRE aufrufen
27D1  E1           POP   HL          ;Programmzeiger laden
27D2  D7           RST   10H         ;nächstes Zeichen adressieren
27D3  C9           RET               ;fertig

                   ************************************

                   FRE - Funktion
                   Größe des freien Speichers oder freien Stringbereichs
                   ermitteln
```

```
27D4  2A FD 78        LD    HL,(78FDH)     ;Anfangsadr. des freien Speichers
27D7  EB              EX    DE,HL          ;in DE
27D8  21 00 00        LD    HL,0           ;Stackpointer in HL
27DC  E7              RST   20H            ;Typ testen. String ?
27DD  20 0D           JR    NZ,27ECH       ;nein!
27DF  CD DA 29        CALL  29DAH          ;Argument aus Zwischenspeicher und
                                           ;Stringbereich löschen
27E2  CD E6 28        CALL  28E6H          ;Stringbereich umsortieren, über-
                                           ;flüssige Strings entfernen
27E5  2A A0 78        LD    HL,(78A0H)     ;Stringbereich-Anfang - 1
27E8  EB              EX    DE,HL          ;in DE
27E9  2A D6 78        LD    HL,(78D6H)     ;letztes freies Byte des Stringber.

27EC  7D              LD    A,L            ;Differenz zwischen HL und DE
27ED  93              SUB   E              ;= Größe des freien Speichers oder
27EE  6F              LD    L,A            ;des freien Stringbereichs
27EF  7C              LD    A,H            ;(MSB)
27F0  9A              SBC   A,D
27F1  67              LD    H,A
27F2  C3 66 0C        JP    0C66H          ;HL mit einf.Gen. in X, fertig

                      ************************************

                      POS - Funktion
                      ermittelt die Cursor-Position
27F5  3A A6 78        LD    A,(78A6H)      ;Cursorposition laden

                      Zahl als Integer (ohne Vorzeichen) in X
27F8  6F              LD    L,A            ;Zahl in L
27F9  AF              XOR   A              ;A = 0
27FA  67              LD    H,A            ;H = 0
27FB  C3 9A 0A        JP    0A9AH          ;HL als Integer in X

                      ************************************

                      USR - Funktion
                      Aufruf einer Maschinenprogramm-Routine
                      Als Parameter werden das Argument in X, der Typ in A
                      und bei Strings die Adresse des Strings in DE übergeben.
27FE  CD A9 79        CALL  79A9H          ;RAM-Erweiterungsausgang
2801  D7              RST   10H            ;nächstes Zeichen im Programm
2802  CD 2C 25        CALL  252CH          ;Argument auswerten (in X)
2805  E5              PUSH  HL             ;Programmzeiger auf den Stack
2806  21 90 08        LD    HL,0890H       ;Rücksprungadresse auf Stack
```

```
2809  E5           PUSH   HL
280A  3A AF 78     LD     A,(78AFH)      ;Typ des Arguments laden
280D  F5           PUSH   AF             ;auf den Stack
280E  FE 03        CP     3              ;= String?
2810  CC DA 29     CALL   Z,29DAH        ;ja, letzten String aus Zwischensp.
                                          ;und Stringbereich entfernen
2813  F1           POP    AF             ;Typ wieder laden
2814  EB           EX     DE,HL          ;Stringadresse in DE
2815  2A 8E 78     LD     HL,(788EH)     ;Startadresse der Masch.-routine
2818  E9           JP     (HL)           ;Routine anspringen

                   ************************************

                   Wert in gewünschten Typ umwandeln
                   Eing.: A = Typ
                          X = Ausgangswert
                   Ausg.: X = Ergebnis im gew. Typ
2819  E5           PUSH   HL             ;HL auf den Stack
281A  E6 07        AND    7              ;Typ=dopp.Gen, 0 als Tab.Offset
281C  21 A1 18     LD     HL,18A1H       ;Sprungtabelle für Typumwandlung
281F  4F           LD     C,A            ;Tab.Offset (=Typ, außer bei dopp.)
2820  06 00        LD     B,0            ;in BC
2822  09           ADD    HL,BC          ;auf Sprungtabellen-Anfang addieren
2823  CD 86 25     CALL   2586H          ;nochmals addieren, Adresse laden
                                          ;und anspringen
2826  E1           POP    HL             ;HL wiederherstellen
2827  C9           RET                   ;fertig

                   ************************************

                   Prüfen, ob Ausführung im DIRECT-Modus.
                   wenn ja, ILLEGAL DIRECT OPERATION - Error
2828  E5           PUSH   HL             ;Programmzeiger auf den Stack
2829  2A A2 78     LD     HL,(78A2H)     ;Aktuelle Zeilennummer laden
282C  23           INC    HL             ;=FFFF ?
282D  7C           LD     A,H            ;(=Direkt-Mode)
282E  B5           OR     L
282F  E1           POP    HL             ;Programmzeiger laden
2830  C0           RET    NZ             ;nein, zurück

2831  1E 16        LD     E,16H          ;Fehlercode in E
2833  C3 A2 19     JP     19A2H          ;ILLEGAL DIREKT OPERATION ausgeben

                   ************************************
```

```
                        STR$ - Funktion
                        Zahl in String umformen
2836  CD BD 0F          CALL   0FBDH          ;Zahl in String umformen
2839  CD 65 28          CALL   2865H          ;String in Zwischenspeicher
                                              ;und X übernehmen
283C  CD DA 29          CALL   29DAH          ;String aus Zwischenspeicher lösch.
283F  01 2B 2A          LD     BC,2A2BH       ;Rücksprungadresse setzen
2842  C5                PUSH   BC
2843  7E                LD     A,(HL)         ;Stringlänge in A
2844  23                INC    HL             ;Stringzeiger + 1
2845  E5                PUSH   HL             ;Stringzeiger auf den Stack
2846  CD BF 28          CALL   28BFH          ;Platz für String im Stringbereich
                                              ;reservieren
2849  E1                POP    HL             ;Stringzeiger laden
284A  4E                LD     C,(HL)         ;Stringadresse laden
284B  23                INC    HL             ;(in BC)
284C  46                LD     B,(HL)
284D  CD 5A 28          CALL   285AH          ;Adresse im Stringbereich in vorl.
                                              ;Zwischenspeicher übertragen
2850  E5                PUSH   HL             ;Zwischenspeicheradresse auf Stack
2851  6F                LD     L,A            ;Stringlänge in L
2852  CD CE 29          CALL   29CEH          ;String in Stringbereich übernehmen
2855  D1                POP    DE             ;Zwischenspeicheradresse laden
2856  C9                RET                   ;Zwischenspeicher in X, fertig

                        ************************************

                        Adresse im Stringbereich ermitteln und im vorläufigen
                        Zwischenspeicher ablegen
                        Eing.: A = Stringlänge
                        Ausg.: DE = Stringadresse im Stringbereich
                               HL = Adresse des vorläufigen Zwischenspeichers
                               Stringlänge+Stringadr. im vorl.Zwischenspeicher
2857  CD BF 28          CALL   28BFH          ;Platz im Stringbereich reservieren
285A  21 D3 78          LD     HL,78D3H       ;Adr. des vorl. Zwischenspeichers
285D  E5                PUSH   HL             ;auf den Stack
285E  77                LD     (HL),A         ;Stringlänge eintragen
285F  23                INC    HL             ;Adr. Zwischenspeicher + 1
2860  73                LD     (HL),E         ;Stringadresse eintragen
2861  23                INC    HL
2862  72                LD     (HL),D
2863  E1                POP    HL             ;Adresse d. Zwischenspeichers laden
2864  C9                RET
```

```
                        ************************************

                        Stringkonstante in Zwischenspeicher und X übernehmen
                        Eing.: HL = Zeiger auf die Stringkonstante
                        Ausg.: Stringlänge und Stringadresse in Zwischen-
                               speicher, Adresse des Zwischenspeicher in X
2865  2B                DEC   HL              ;Stringzeiger - 1
2866  06 22             LD    B,22H           ;Trennzeichen 1 = '"'
2868  50                LD    D,B             ;= Trennzeichen 2
2869  E5                PUSH  HL              ;Stringzeiger - 1 auf Stack
286A  0E FF             LD    C,0FFH          ;Zeichenzähler = -1
286C  23                INC   HL              ;Stringzeiger + 1
286D  7E                LD    A,(HL)          ;Zeichen laden
286E  0C                INC   C               ;Zeichenzähler + 1
286F  B7                OR    A               ;Zeilenende ?
2870  28 06             JR    Z,2878H         ;ja, Stringende
2872  BA                CP    D               ;= Trennzeichen 2 ?
2873  28 03             JR    Z,2878H         ;ja, Stringende
2875  B8                CP    B               ;= Trennzeichen 1 ?
2876  20 F4             JR    NZ,286CH        ;nein, nächstes Zeichen
2878  FE 22             CP    '"'             ;letztes Zeichen '"' ?
287A  CC 78 1D          CALL  Z,1D78H         ;ja, nächstes Zeichen
287D  E3                EX    (SP),HL         ;Stringzeiger - 1 laden,
                                              ;derz. Stringzeiger auf Stack
287E  23                INC   HL              ;Stringzeiger + 1
287F  EB                EX    DE,HL           ;in DE
2880  79                LD    A,C             ;Stringlänge in A
2881  CD 5A 28          CALL  285AH           ;String in vorl. Zwischenspeicher
2884  11 D3 78          LD    DE,78D3H        ;Adr. des vorl. Zwischenspeichers
2887  3E                DEFB  3EH             ;LD A,0D5H  Dummy-Befehl
2888  D5                POP   DE              ;Stringzeiger in DE laden
2889  2A B3 78          LD    HL,(78B3H)      ;Momentane Zwischensp.adr. laden
288C  22 21 79          LD    (7921H),HL      ;in X eintragen
288F  3E 03             LD    A,3             ;Typ = String setzen
2891  32 AF 78          LD    (78AFH),A
2894  CD D3 09          CALL  09D3H           ;Vorläufigen Zwischenspeicher in
                                              ;nächsten Zwischenspeicherplatz
2897  11 D6 78          LD    DE,78D6H        ;Zwischenspeicher voll?
289A  DF                RST   18H
289B  22 B3 78          LD    (78B3H),HL      ;nächste Zwischenspeicheradr. merk.
289E  E1                POP   HL              ;Programmzeiger laden
289F  7E                LD    A,(HL)          ;nächstes Zeichen laden
28A0  C0                RET   NZ              ;Zw.Sp. nicht voll, fertig
```

```
                        STRING FORMULA TOO COMPLEX - Error
28A1  1E 1E             LD      E,1EH           ;Fehlercode in E
28A3  C3 A2 19          JP      19A2H           ;Fehlermeldung ausgeben

                        ************************************

                        String ausdrucken
                        Eing.: HL = Stringadresse
                               String durch '"' oder 00 abgeshlossen
28A6  23                INC     HL              ;Stringadresse + 1
28A7  CD 65 28          CALL    2865H           ;String in Zwischenspeicher + X
28AA  CD DA 29          CALL    29DAH           ;String aus Zwischenspeicher lösch.
28AD  CD C4 09          CALL    09C4H           ;Stringadr. in BC, Länge in D
28B0  14                INC     D               ;Stringlänge + 1
28B1  15                DEC     D               ;alle Zeichen ausgegeben?
28B2  C8                RET     Z               ;ja, fertig
28B3  0A                LD      A,(BC)          ;Zeichen laden
28B4  CD 2A 03          CALL    032AH           ;und ausgeben
28B7  FE 0D             CP      0DH             ;Carriage Return ?
28B9  CC 03 21          CALL    Z,2103H         ;ja, über RAM-Ausgang 79D0H zur.
28BC  03                INC     BC              ;Stringadresse + 1
28BD  18 F2             JR      28B1H           ;nächstes Zeichen

                        ************************************

                        Platz für einen String im Stringbereich reservieren
                        Eing.: A = Stringlänge
                        Ausg.: DE = Adresse im Stringbereich
28BF  B7                OR      A               ;PACK-Flag löschen
28C0  0E                DEFB    0EH             ;LD C,0F1H  Dummy-Befehl
28C1  F1                POP     AF              ;PACK-Flag vom Stack laden
28C2  F5                PUSH    AF              ;auf den Stack legen
28C3  2A A0 78          LD      HL,(78A0H)      ;Anfang des Stringbereichs - 1
28C6  EB                EX      DE,HL           ;in DE
28C7  2A D6 78          LD      HL,(78D6H)      ;Zeiger auf 1.freies Byte im Strber
28CA  2F                CPL                     ;Stringlänge komplementieren
28CB  4F                LD      C,A             ;und in BC
28CC  06 FF             LD      B,0FFH
28CE  09                ADD     HL,BC           ;Stringbereichs-Zeiger - Länge
28CF  23                INC     HL              ;+ 1 (Ergebniskorrektur)
28D0  DF                RST     18H             ;< Anfang des Stringbereichs - 1?
28D1  38 07             JR      C,28DAH         ;ja, Stringbereich packen
28D3  22 D6 78          LD      (78D6H),HL      ;neuen Stringbereichszeiger speich.
```

```
28D6  23           INC   HL            ;+ 1 = Stringadresse im Str.bereich
28D7  EB           EX    DE,HL         ;in DE übertragen
28D8  F1           POP   AF            Stringlänge wieder laden
28D9  C9           RET                 ;fertig

                   ************************************

                   Stringbereich packen
28DA  F1           POP   AF            ;PACK-Flag laden
28DB  1E 1A        LD    E,1AH         ;Fehlercode in E
28DD  CA A2 19     JP    Z,19A2H       ;bereits gepackt, OUT OF STRING SPC
28E0  BF           CP    A             ;PACK-Flag setzen
28E1  F5           PUSH  AF            ;und auf den Stack
28E2  01 C1 28     LD    BC,28C1H      ;Rücksprungadresse setzen
28E5  C5           PUSH  BC            ;nach dem Packen erneut versuchen,
                                       ;ob genügend Platz vorhanden.
28E6  2A B1 78     LD    HL,(78B1H)    ;Stringbereichszeiger = RAM-Endadr.
28E9  22 D6 78     LD    (78D6H),HL
28EC  21 00 00     LD    HL,0          ;höchster Stringzeiger = 0
28EF  E5           PUSH  HL            ;auf den Stack
28F0  2A A0 78     LD    HL,(78A0H)    ;höchster String = Anf Stringber.
28F3  E5           PUSH  HL            ;auf den Stack '

                   Höchsten String unter Zwischenspeicher-Strings suchen
28F4  21 B5 78     LD    HL,78B5H      ;Anfang Zwischenspeicher in HL
28F7  EB           EX    DE,HL
28F8  2A B3 78     LD    HL,(78B3H)    ;Adr. nächster freier Zwischen-
28FB  EB           EX    DE,HL         ;speicherplatz in DE
28FC  DF           RST   18H           ;Zwischenspeicher abgearbeitet?
28FD  01 F7 28     LD    BC,28F7H      ;Rücksprungadresse für
                                       ;nächsten Zwischenspeicherpl. laden
2900  C2 4A 29     JP    NZ,294AH      ;nein, höchsten String aktualis.

                   Höchsten String unter einfachen Variablen suchen
2903  2A F9 78     LD    HL,(78F9H)    ;Anfadr. der Var.Tabelle in HL
2906  EB           EX    DE,HL         ;Endadr. der Var.Tabelle in DE
2907  2A FB 78     LD    HL,(78FBH)
290A  EB           EX    DE,HL
290B  DF           RST   18H           ;Ende der Variablen-Tabelle?
290C  28 13        JR    Z,2921H       ;Ja, Matrizen untersuchen
290E  7E           LD    A,(HL)        ;Typ aus Var.Tabelle laden
290F  23           INC   HL            ;Var.Tab.-Adresse auf Wert
2910  23           INC   HL
2911  23           INC   HL
```

```
2912  FE 03          CP    3            ;Stringvariable?
2914  20 04          JR    NZ,291AH     ;nein!
2916  CD 4B 29       CALL  294BH        ;höchsten String aktualisieren
2919  AF             XOR   A            ;A = 0,damit Zeiger nicht erh.wird
291A  5F             LD    E,A          ;Typ in DE übertragen
291B  16 00          LD    D,0
```

```
291D  19           ADD   HL,DE          ;Adresse Var.Tab. + Typ (Länge)
291E  18 E6        JR    2906H          ;nächsten Eintrag

                   Höchsten String in Matrix-Tabelle suchen
2920  C1           POP   BC             ;Stack korrigieren
2921  EB           EX    DE,HL
2922  2A FD 78     LD    HL,(78FDH)     ;Endadr. Matrix-Tabelle in DE
2925  EB           EX    DE,HL
2926  DF           RST   18H            ;Ende erreicht ?
2927  CA 6B 29     JP    Z,296BH        ;ja, höchsten String an nächst
                                        ;höhere Stelle im Stringbereich
292A  7E           LD    A,(HL)         ;Typ der Matrix laden
292B  23           INC   HL             ;Tabellenadresse + 1
292C  CD C2 09     CALL  09C2H          ;Matrixlänge in BC
                                        ;Tabellenadresse auf DIM-Zähler
292F  E5           PUSH  HL             ;Tabellenadresse auf Stack
2930  09           ADD   HL,BC          ;+Matrixlänge = Anfang der n.Matrix
2931  FE 03        CP    3              ;String-Matrix ?
2933  20 EB        JR    NZ,2920H       ;nein, nächste Matrix
2935  22 D8 78     LD    (78D8H),HL     ;Adr. der n. Matrix speichern
2938  E1           POP   HL             ;Adresse DIM-Zähler laden
2939  4E           LD    C,(HL)         ;DIM-Zähler laden
293A  06 00        LD    B,0
293C  09           ADD   HL,BC          ;auf Adresse DIM-Zähler 2* addieren
293D  09           ADD   HL,BC
293E  23           INC   HL             ;+ 1 = Wertadresse der Matrix
293F  EB           EX    DE,HL
2940  2A D8 78     LD    HL,(78D8H)     ;Adr. der n. Matrix in DE
2943  EB           EX    DE,HL
2944  DF           RST   18H            ;Matrix vollständig bearbeitet?
2945  28 DA        JR    Z,2921H        ;ja, nächste Matrix
2947  01 3F 29     LD    BC,293FH       ;nein,Rücksprungadresse laden

                   String mit bis jetzt höchstem String vergleichen und,
                   wenn im Stringbereich höher, diesen ersetzen.
294A  C5           PUSH  BC             ;Rücksprungadresse auf Stack
294B  AF           XOR   A              ;Stringlänge = 0 ?
294C  B6           OR    (HL)
294D  23           INC   HL             ;Stringzeiger auf Stringadresse
294E  5E           LD    E,(HL)         ;Stringadresse laden
294F  23           INC   HL
2950  56           LD    D,(HL)
2951  23           INC   HL             ;Stringzeiger + 1
2952  C8           RET   Z              ;Stringlänge = 0, fertig!
```

```
2953  44           LD    B,H          ;Stringzeiger in BC
2954  4D           LD    C,L
2955  2A D6 78     LD    HL,(78D6H)   ;Stringadresse > Stringber.-Zeiger?
2958  DF           RST   18H          ;(String schon neu einsortiert)
2959  60           LD    H,B          ;Stringzeiger wieder in HL
295A  69           LD    L,C
295B  D8           RET   C            ;ja, fertig!
295C  E1           POP   HL           ;Rücksprungadresse in HL
295D  E3           EX    (SP),HL      ;Adresse des höchsten Strings laden
                                      ;Rücksprungadresse wieder auf Stack
295E  DF           RST   18H          ;zu untersuchender String höher als
                                      ;höchster String ?
295F  E3           EX    (SP),HL      ;Adresse des höchsten Strings auf
                                      ;Stack, Rücksprungadresse laden
2960  E5           PUSH  HL           ;Rücksprungadresse wieder auf Stack
2961  60           LD    H,B          ;Stringzeiger wieder in HL
2962  69           LD    L,C
2963  D0           RET   NC           ;nein, fertig!
2964  C1           POP   BC           ;Rücksprungadresse in BC
2965  F1           POP   AF           ;Adresse und Zeiger des höchsten
2966  F1           POP   AF           ;Strings vom Stack holen
2967  E5           PUSH  HL           ;Zeiger und Adresse des gerade
2968  D5           PUSH  DE           ;untersuchten Strings als neuen
                                      ;höchsten String auf Stack
2969  C5           PUSH  BC           ;Rücksprungadresse auf Stack
296A  C9           RET                ;fertig

                   Höchsten String einsortieren
296B  D1           POP   DE           ;Adresse des höchsten Strings laden
296C  E1           POP   HL           ;Zeiger des höchsten Strings in HL
296D  7D           LD    A,L          ;Zeiger = 0 ?
296E  B4           OR    H            ;(alle Strings einsortiert)
296F  C8           RET   Z            ;ja, fertig!
2970  2B           DEC   HL           ;Stringzeiger auf Stringadresse
2971  46           LD    B,(HL)       ;Stringadresse laden
2972  2B           DEC   HL
2973  4E           LD    C,(HL)
2974  E5           PUSH  HL           ;Stringzeiger auf den Stack
2975  2B           DEC   HL           ;Stringlänge laden
2976  6E           LD    L,(HL)       ;in HL
2977  26 00        LD    H,0
2979  09           ADD   HL,BC        ;+ Stringadresse
297A  50           LD    D,B          ;Stringadresse in DE
297B  59           LD    E,C
```

```
297C  2B           DEC   HL          ;HL = Stringende
297D  44           LD    B,H         ;in BC
297E  4D           LD    C,L
297F  2A D6 78     LD    HL,(78D6H)  ;Zeiger auf Stringbereich laden
2982  CD 58 19     CALL  1958H       ;String unterhalb des String-
                                     ;bereich-Zeigers abpeichern
2985  E1           POP   HL          ;Stringzeiger wieder laden
2986  71           LD    (HL),C      ;neue Stringadresse speichern
2987  23           INC   HL
2988  70           LD    (HL),B
2989  69           LD    L,C         ;und in HL
298A  60           LD    H,B
298B  2B           DEC   HL          ;- 1 = neuer Stringbereich-Zeiger
298C  C3 E9 28     JP    28E9H       ;abspeichern und weiter

                   ************************************

                   String-Verknüpfung
298F  C5           PUSH  BC          ;Letzte Priorität auf Stack
2990  E5           PUSH  HL          ;Programmzeiger auf Stack
2991  2A 21 79     LD    HL,(7921H)  ;1. Stringzeiger in HL
2994  E3           EX    (SP),HL     ;Programmzeiger laden
                                     ;1. Stringzeiger auf Stack
2995  CD 9F 24     CALL  249FH       ;2. Operanden bestimmen
2998  E3           EX    (SP),HL     ;Programmzeiger auf Stack
                                     ;1. Stringzeiger laden
2999  CD F4 0A     CALL  0AF4H       ;2. Operand String? nein - TM-Error
299C  7E           LD    A,(HL)      ;1. Stringlänge laden
299D  E5           PUSH  HL          ;1. Stringzeiger auf den Stack
299E  2A 21 79     LD    HL,(7921H)  ;2. Stringzeiger laden
29A1  E5           PUSH  HL          ;und auch auf den Stack
29A2  86           ADD   (HL)        ;Stringlängen addieren
29A3  1E 1C        LD    E,1CH       ;Fehlercode in E
29A5  DA A2 19     JP    C,19A2H     ;> 256? ja-STRING TOO LONG - Error
29A8  CD 57 28     CALL  2857H       ;Platz für verknüpften String
                                     ;schaffen, in vorl.Zwischensp.eintr
29AB  D1           POP   DE          ;2. Stringzeiger wieder laden
29AC  CD DE 29     CALL  29DEH       ;2. String aus Zwischensp.austragen
29AF  E3           EX    (SP),HL     ;2. Stringzeiger auf Stack
                                     ;1. Stringzeiger laden
29B0  CD DD 29     CALL  29DDH       ;1. String aus Zwischensp.entfernen
29B3  E5           PUSH  HL          ;1. Stringzeiger auf den Stack
29B4  2A D4 78     LD    HL,(78D4H)  ;Stringadr. aus vorl.Zw.speicher l.
29B7  EB           EX    DE,HL       ;in DE
```

```
29B8  CD C6 29        CALL   29C6H          ;1. String in Stringbereich übertr.
29BB  CD C6 29        CALL   29C6H          ;2. String in Stringbereich übertr.
29BE  21 49 23        LD     HL,2349H       ;Rücksprungadresse laden
29C1  E3              EX     (SP),HL        ;mit Programmzeigeer tauschen
29C2  E5              PUSH   HL             ;Programmzeiger wieder auf Stack
29C3  C3 84 28        JP     2884H          ;Vorl.Zwischenspeicher in X und
                                            ;Zwischenspeicher übertragen

                      ************************************

                      String in Stringbereich übertragen
29C6  E1              POP    HL             ;Rücksprungadresse laden
29C7  E3              EX     (SP),HL        ;Stringzeiger laden,
                                            ;Rücksprungadresse auf Stack
29C8  7E              LD     A,(HL)         ;Stringlänge laden
29C9  23              INC    HL             ;Stringzeiger + 1
29CA  4E              LD     C,(HL)         ;Stringadresse laden
29CB  23              INC    HL
29CC  46              LD     B,(HL)
29CD  6F              LD     L,A            ;Stringlänge in L
29CE  2C              INC    L              ;+ 1
29CF  2D              DEC    L              ;Stringlänge - 1, = 0 ?
29D0  C8              RET    Z              ;ja, fertig!
29D1  0A              LD     A,(BC)         ;Ein Zeichen in Stringbereich
29D2  12              LD     (DE),A
29D3  03              INC    BC             ;Stringzeiger + 1
29D4  13              INC    DE             ;Adresse Stringbereich + 1
29D5  18 F8           JR     29CFH          ;nächstes Byte

                      ************************************

                      String aus Zwischenspeicher und Stringbereich entfernen


29D7  CD F4 0A        CALL   0AF4H          ;kein String in X? ja-TYPE MISMATCH
29DA  2A 21 79        LD     HL,(7921H)     ;Stringzeiger aus X laden
29DD  EB              EX     DE,HL          ;in DE
29DE  CD F5 29        CALL   29F5H          ;String zuoberst im Zw.speicher?
                                            ;ja - entfernen!
29E1  EB              EX     DE,HL          ;Stringzeiger in HL
29E2  C0              RET    NZ             ;nicht entfernt? fertig!

                      String aus Stringbereich entfernen
```

```
29E3  D5           PUSH  DE            ;Stringzeiger auf den Stack
29E4  50           LD    D,B           ;Stringadresse in DE
29E5  59           LD    E,C
29E6  1B           DEC   DE            ;- 1
29E7  4E           LD    C,(HL)        ;Stringlänge in C
29E8  2A D6 78     LD    HL,(78D6H)    ;Stringbereichs-Zeiger in HL
29EB  DF           RST   18H           ;= Stringadresse - 1 ?
29EC  20 05        JR    NZ,29F3H      ;nein, fertig!
29EE  47           LD    B,A           ;B = 0
29EF  09           ADD   HL,BC         ;Stringbereichs-zeiger + Länge
29F0  22 D6 78     LD    (78D6H),HL    ;= neuer Stringsbereichs-Zeiger
29F3  E1           POP   HL            ;Stringzeiger laden
29F4  C9           REET

                   ************************************

                   String aus Zwischenspeicher entfernen
29F5  2A B3 78     LD    HL,(78B3H)    ;nächste Zwischenspeicheradr.laden
29F8  2B           DEC   HL            ;- 1
29F9  46           LD    B,(HL)        ;Adresse des letzten Strings laden
29FA  2B           DEC   HL
29FB  4E           LD    C,(HL)
29FC  2B           DEC   HL            ;Zeiger auf Anfang des letzt.Eintr.
29FD  DF           RST   18H           ;= Stringzeiger
29FE  C0           RET   NZ            ;nein!
29FF  22 B3 78     LD    (78B3H),HL    ;neuen Zeiger abspeichern,
                                       ;letzter Eintrag ist gelöscht
2A02  C9           RET

                   ************************************

                   LEN - Funktion
                   ermittelt die Länge eines Strings
2A03  01 F8 27     LD    BC,27F8H      ;Rücksprungadresse setzen
2A06  C5           PUSH  BC
2A07  CD D7 29     CALL  29D7H         ;Argument-String aus Zwischen-
                                       ;speicher + Stringbereich entfern.
2A0A  AF           XOR   A             ;D = 0
2A0B  57           LD    D,A
2A0C  7E           LD    A,(HL)        ;Stringlänge laden
2A0D  B7           OR    A             ;und testen
2A0E  C9           RET                 ;weiter bei 27F8H

                   ************************************
```

```
                        ASC - Funktion
                        ermittelt ASCII-Code des 1. Zeichens eines Strings
2A0F  01 F8 27          LD      BC,27F8H        ;Rücksprungadresse setzen
2A12  C5                PUSH    BC
2A13  CD 07 2A          CALL    2A07H           ;Stringlänge = 0 ?
2A16  CA 4A 1E          JP      Z,1E4AH         ;ja, FUNCTION CODE - Error
2A19  23                INC     HL              ;Stringzeiger auf Stringadresse
2A1A  5E                LD      E,(HL)          ;Stringadresse laden
2A1B  23                INC     HL
2A1C  56                LD      D,(HL)
2A1D  1A                LD      A,(DE)          ;1. Zeichen laden
2A1E  C9                RET                     ;weiter bei 27F8H

                        ************************************

                        CHR$ - Funktion
                        Erzeugt aus dem Argument (ASCII-Code) einen 1-Byte String
2A1F  3E 01             LD      A,1             ;Stringlänge = 1
2A21  CD 57 28          CALL    2857H           ;Platz im Stringbereich reservieren
                                                ;und in vorl.Zwischenspeicher eintr
2A24  CD 1F 2B          CALL    2B1FH           ;Ganzz.Wert des Arguments in E
2A27  2A D4 78          LD      HL,(78D4H)      ;Stringadr. aus vorl.Zw.speicher 1.
2A2A  73                LD      (HL),E          ;Zeichen dort abspeichern
2A2B  C1                POP     BC              ;Rücksprungadresse entfernen
2A2C  C3 84 28          JP      2884H           ;Vorl.Zwischenspeicher nach X und
                                                ;in Zwischenspeicher übertragen

                        ************************************

                        STRING$ - Funktion
                        erzeugt einen String aus n gleichen Zeichen
2A2F  D7                RST     10H             ;nächstes Zeichen
2A30  CF                RST     8               ;folgt ein '(' ?
2A31  28                DEFB    '('
2A32  CD 1C 2B          CALL    2B1CH           ;Stringlänge auswerten und in E
2A35  D5                PUSH    DE              ;Stringlänge auf Stack
2A36  CF                RST     8               ;folgt ein Komma ?
2A37  2C                DEFB    ','
2A38  CD 37 23          CALL    2337H           ;Zeichenausdruck auswerten
2A3B  CF                RST     8               ;folgt ein ')' ?
2A3C  29                DEFB    ')'
2A3D  E3                EX      (SP),HL         ;Stringlänge in L laden
                                                ;Programmzeiger auf den Stack
```

```
2A3E  E5           PUSH  HL          ;Stringlänge wieder auf den Stack
2A3F  E7           RST   20H         ;Zeichenausdruck = String ?
2A40  28 5         JR    Z,2A47H     ;ja!
2A42  CD 1F 2B     CALL  2B1FH       ;nein, ganzz. Wert in A
2A45  18 03        JR    2A4AH       ;weiter bei 2A4AH
2A47  CD 13 2A     CALL  2A13H       ;1. Zeichen des Strings in A
2A4A  D1           POP   DE          ;Stringlänge in E
2A4B  F5           PUSH  AF          ;Zeichen 2x auf Stack
2A4C  F5           PUSH  AF
2A4D  7B           LD    A,E         ;Platz im Stringbereich reservieren
2A4E  CD 57 28     CALL  2857H       ;String in vorl.Zwischenspeicher
2A51  5F           LD    E,A         ;Stringlänge in E
2A52  F1           POP   AF          ;Zeichen wieder laden
2A53  1C           INC   E           ;Stringlänge = 0 ?
2A54  1D           DEC   E
2A55  28 D4        JR    Z,2A2BH     ;ja, vorl.Zw.sp. in Zw.speicher + X
2A57  2A D4 78     LD    HL,(78D4H)  ;Stringadresse laden
2A5A  77           LD    (HL),A      ;Zeichen übertragen
2A5B  23           INC   HL          ;Stringbereichs-Adresse + 1
2A5C  1D           DEC   E           ;Stringlänge - 1, = 0 ?
2A5D  20 FB        JR    NZ,2A5AH    ;nein, nächstes Zeichen!
2A5F  18 CA        JR    2A2B        ;vorl.Zw.sp. in Zw.speicher u. X

                   ************************************

                   LEFT$ - Funktion
                   linken Teil eines Strings abtrennen
2A61  CD DF 2A     CALL  2ADFH       ;Programmzeiger in HL, ')' testen,
                                     ;2. Argument in B
2A64  AF           XOR   A           ;Left-Offset = 0 (f.RIGHT$ u.MID$)
2A65  E3           EX    (SP),HL     ;Programmzeiger auf Stack
                                     ;Stringzeiger laden
2A66  4F           LD    C,A         ;Left-Offset in C
2A67  3E           DEFB  3EH         ;LD A,0E5H  Dummy-Befehl
2A68  E5           PUSH  HL          ;Stringzeiger auf Stack
                                     ;(Stackkorrektur bei USING)
2A69  E5           PUSH  HL          ;Stringzeiger auf Stack
2A6A  7E           LD    A,(HL)      ;Stringlänge laden
2A6B  B8           CP    B           ;< 2. Argument?
2A6C  38 02        JR    C,2A70H     ;ja, Erg.stringl.=Stringlänge
2A6E  78           LD    A,B         ;nein, Erg.stringlänge = 2.Argument
2A6F  11           DEFB  11H         ;LD DE,000EH  Dummy-Befehl
2A70  0E 00        LD    C,0         ;Left-Offset = 0
2A72  C5           PUSH  BC          ;Left-Offset auf Stack
```

```
2A73  CD BF 28        CALL   28BFH          ;Platz f.Ergebnis im Stringber.res.
2A76  C1              POP    BC             ;Left-Offset wieder laden
2A77  E1              POP    HL             ;Stringzeiger laden
2A78  E5              PUSH   HL             ;und wieder auf den Stack
2A79  23              INC    HL             ;Stringzeiger auf Stringadresse
2A7A  46              LD     B,(HL)         ;Stringadresse in HL laden
2A7B  23              INC    HL
2A7C  66              LD     H,(HL)         ;(MSB)
2A7D  68              LD     L,B            ;(LSB)
2A7E  06 00           LD     B,0            ;Left-Offset auf Stringadr.addieren
2A80  09              ADD    HL,BC
2A81  44              LD     B,H            ;Ergebnis-Stringadresse in BC
2A82  4D              LD     C,L
2A83  CD 5A 28        CALL   285AH          ;Ergebnis-String in vorl.Zw.speich.
2A86  6F              LD     L,A            ;Erg.Stringlänge in L
2A87  CD CE 29        CALL   29CEH          ;Ergebnis-String in Stringbereich
2A8A  D1              POP    DE             ;Stringzeiger in DE
2A8B  CD DE 29        CALL   29DEH          ;Argument-String aus Zw.speicher
                                            ;und Stringbereich löschen
2A8E  C3 84 28        JP     2884H          ;Vorl.Zw.speicher in Zw.speicher +X

                      ************************************

                      RIGHT$ - Funktion
                      rechten Teil eines Strings abtrennen
2A91  CD DF 2A        CALL   2ADFH          ;Programmzeiger in HL, ')' testen,
                                            ;2. Argument in B
2A94  D1              POP    DE             ;Stringzeiger laden
2A95  D5              PUSH   DE             ;und wieder auf Stack
2A96  1A              LD     A,(DE)         ;Stringlänge laden
2A97  90              SUB    B              ;- 2. Argument = Left-Offset
2A98  18 CB           JR     2A65H          ;weiter bei LEFT$

                      ************************************

                      MID$ - Funktion
                      mittleren Teil eines Strings abtrennen
2A9A  EB              EX     DE,HL          ;Programmzeiger in HL
2A9B  7E              LD     A,(HL)         ;Zeichen laden
2A9C  CD E2 2A        CALL   2AE2H          ;2. Argument in B
2A9F  04              INC    B              ;2. Argument = 0 ?
2AA0  05              DEC    B
2AA1  CA 4A 1E        JP     Z,1E4AH        ;ja, FUNCTION CODE - Error
2AA4  C5              PUSH   BC             ;2. Argument auf den Stack
```

```
2AA5  1E FF       LD    E,0FFH      ;3. Argument = 255 (Std.=Reststring
2AA7  FE 29       CP    ')'         ;')' als Abschluß der Parameter?
2AA9  28 05       JR    Z,2AB0H     ;ja, nur 2 Argumente
2AAB  CF          RST   8           ;nein, folgt ein Komma ?
2AAC  2C          DEFB  ','
2AAD  CD 1C 2B    CALL  2B1CH       ;3. Argument auswerten (in E)
2AB0  CF          RST   8           ;jetzt muß ')' folgen
2AB1  29          DEFB  ')'
2AB2  F1          POP   AF          ;2. Argument in A laden
2AB3  E3          EX    (SP),HL     ;Programmzeiger auf Stack
                                    ;Stringzeiger laden
2AB4  01 69 2A    LD    BC,2A69H    ;Rücksprungadresse setzen
2AB7  C5          PUSH  BC          ;(Sprung in LEFT$)
2AB8  3D          DEC   A           ;2. Arg. - 1 = Left-Offset
2AB9  BE          CP    (HL)        ;Stringlänge < 2. Argument ?
2ABA  06 00       LD    B,0         ;Ergebnis-Stringlänge = 0
2ABC  D0          RET   NC          ;ja, Ergebnis-String ist leer
2ABD  4F          LD    C,A         ;Left-Offset in C
2ABE  7E          LD    A,(HL)      ;Stringlänge laden
2ABF  91          SUB   C           ;- Left-Offset
2AC0  B8          CP    E           ;< 3. Argument ?
2AC1  47          LD    B,A         ;Erg.Stringlänge = Differenz
2AC2  D8          RET   C           ;ja, ges.Reststring=Ergebnis-String
2AC3  43          LD    B,E         ;nein, 3. Argument = Erg.stringlng.
2AC4  C9          RET               ;weiter bei LEFT$

                  ************************************

                  VAL - Funktion
                  String in Zahl umwandeln
2AC5  CD 07 2A    CALL  2A07H       ;Stringlänge Argument = 0 ?
2AC8  CA F8 27    JP    Z,27F8H     ;ja, 0 als Integer in X, fertig
2ACB  5F          LD    E,A         ;Stringlänge in E
2ACC  23          INC   HL          ;Stringzeiger + 1
2ACD  7E          LD    A,(HL)      ;Stringadresse in HL
2ACE  23          INC   HL
2ACF  66          LD    H,(HL)
2AD0  6F          LD    L,A
2AD1  E5          PUSH  HL          ;Stringadresse auf den Stack
2AD2  19          ADD   HL,DE       ;+ Stringlänge
2AD3  46          LD    B,(HL)      ;1. Zeichen des nächsten Strings
2AD4  72          LD    (HL),D      ;durch 00 (Zeilenende) ersetzen
2AD5  E3          EX    (SP),HL     ;Stringadr. n.String auf den Stack
                                    ;Stringadresse akt.String laden
```

```
2AD6  C5           PUSH  BC          ;1. Zeichen d.n.Strings sichern
2AD7  7E           LD    A,(HL)      ;1. Zeichen des Strings laden
2AD8  CD 65 0E     CALL  0E65H       ;String in Zahl umwandeln (in X)
2ADB  C1           POP   BC          ;1. Zeichen d.n.Strings laden
2ADC  E1           POP   HL          ;Stringadresse d.n.Strings laden
2ADD  70           LD    (HL),B      ;1. Zeichen in nächsten String zur.
2ADE  C9           RET

                   ************************************

                   Unterprogramm für LEFT$, RIGHT$ und MID$
2ADF  EB           EX    DE,HL       ;Ansprung LEFT$ und RIGHT$
                                     ;Programmzeiger in HL
2AE0  CF           RST   8           ;prüfen, ob mit ')' abgeschlossen
2AE1  29           DEFB  ')'
2AE2  C1           POP   BC          ;Ansprung MID$
                                     ;Rücksprungadresse laden
2AE3  D1           POP   DE          ;2. Argument in E laden
2AE4  C5           PUSH  BC          ;Rücksprungadresse auf Stack
2AE5  43           LD    B,E         ;2. Argument in B
2AE6  C9           RET

                   ************************************

                   Funktions-Token auf der linken Seite einer Zuweisung
2AE7  FE 7A        CP    7AH         ;= MID$ - Token
2AE9  C2 97 19     JP    NZ,1997H    ;nein, SYNTAX ERROR
2AEC  C3 D9 79     JP    79D9H       ;ja, zum RAM-Erweiterungsausgang

                   ************************************

                   INP - Funktion
                   lesen Daten vom Eingabe-Port
2AEF  CD 1F 2B     CALL  2B1FH       ;Ganzz. Wert des Arguments in A
2AF2  32 94 78     LD    (7894H),A   ;in RAM-Unterprogramm als PORT#
2AF5  CD 93 78     CALL  7893H       ;RAM-Unterprogramm aufrufen
2AF8  C3 F8 27     JP    27F8H       ;A-Inhalt als Ergebnis in X

                   ************************************

                   OUT - Anweisung
                   Daten über Ausgabe-Port ausgeben
2AFB  CD 0E 2B     CALL  2B0EH       ;beide Argumente analysieren und
                                     ;Port-Nummer in RAM-Unterprogramm
```

```
2AFE  C3 96 78        JP      7896H           ;RAM-Unterprogramm aufrufen

                      ************************************

                      Ausdruck auswerten und Ergebnis in Integer umwandeln
                      Eing.: HL = Adresse des Ausdrucks im Programm
                      Ausg.: DE = Ergebnis
                             Flags: S = 1  Ergebnis < 0
                                    Z = 1  Ergebnis < 256
2B01  D7              RST     10H             ;nächstes Zeichen adressieren
2B02  CD 37 23        CALL    2337H           ;Ausdruck auswerten
2B05  E5              PUSH    HL              ;Programmzeiger auf Stack
2B06  CD 7F 0A        CALL    0A7FH           ;Ergebnis in Integer umwandeln
2B09  EB              EX      DE,HL           ;Ergebnis in DE
2B0A  E1              POP     HL              ;Programmzeiger laden
2B0B  7A              LD      A,D             ;Flags setzen
2B0C  B7              OR      A
2B0D  C9              RET

                      ************************************

                      2 Argumente für OUT analysieren
2B0E  CD 1C 2B        CALL    2B1CH           ;Port-Nummer übernehmen (in A)
2B11  32 94 78        LD      (7894H),A       ;in INP- und OUT-Unterprogramm
2B14  32 97 78        LD      (7897H),A
2B17  CF              RST     8               ;folgt ein Komma ?
2B18  2C              DEFB    ','
2B19  18 01           JR      2B1CH           ;Wert analysieren (<256) und in A

                      ************************************

                      Ausdruck auswerten, Ergebnis in Integer umwandeln (<256)
2B1B  D7              RST     10H             ;nächstes Zeichen adressieren
2B1C  CD 37 23        CALL    2337H           ;Ausdruck auswerten
2B1F  CD 05 2B        CALL    2B05H           ;Ergebnis in Integer umwandeln (DE)
2B22  C2 4A 1E        JP      NZ,1E4AH        ;> 256 = FUNCTION CODE - Error
2B25  2B              DEC     HL              ;Programmzeiger - 1
2B26  D7              RST     10H             ;nächstes Zeichen adressieren
2B27  7B              LD      A,E             ;Ergebnis in A
2B28  C9              RET

                      ************************************

                      LLIST - Anweisung
```

```
                        Programm auf Drucker auflisten
2B29  3E 01             LD      A,1             ;Ausgabe-Flag auf Drucker
2B2B  32 9C 78          LD      (789CH),A

                        ************************************

                        LIST - Anweisung
                        Programm auf Bildschirm auflisten
2B2E  C1                POP     BC              ;Rücksprungadresse vom Stack
2B2F  CD 10 1B          CALL    1B10H           ;beide Argumente analysieren
                                                ;1.Zeilenadr.in BC, 2.Zeil.nr=Stack
2B32  C5                PUSH    BC              ;1. Zeilenadresse auf Stack
2B33  CD 25 3B          CALL    3B25H           ;Liste unterbrechen/abbrechen?
2B36  22 A2 78          LD      (78A2H),HL      ;Direktbefehl setzen (Znr=FFFF)
2B39  E1                POP     HL              ;1.Zeilenadresse in HL
2B3A  D1                POP     DE              ;2.Zeilennummer in DE
2B3B  4E                LD      C,(HL)          ;Zeilenzeiger der Zeile laden
2B3C  23                INC     HL
2B3D  46                LD      B,(HL)
2B3E  23                INC     HL              ;Zeilenadresse auf Zeilennummer
2B3F  78                LD      A,B             ;Programmende ? (Zeiger=0000)
2B40  B1                OR      C
2B41  CA 19 1A          JP      Z,1A19H         ;ja, zurück zur Hauptschleife
2B44  CD DF 79          CALL    79DFH           ;RAM-Erweiterungsausgang
2B47  CD 9B 1D          CALL    1D9BH           ;Tastenbetätigung auswerten
2B4A  C5                PUSH    BC              ;Adresse der nächst.Zeile auf Stack
2B4B  4E                LD      C,(HL)          ;Zeilennummer laden
2B4C  23                INC     HL
2B4D  46                LD      B,(HL)
2B4E  23                INC     HL              ;Programmzeiger auf Zeilentext
2B4F  C5                PUSH    BC              ;Zeilennummer auf Stack
2B50  E3                EX      (SP),HL         ;Programmzeiger auf den Stack
                                                ;Zeilennummer in HL laden
2B51  EB                EX      DE,HL           ;End-ZNr. in HL, Zeilennummer in DE
2B52  DF                RST     18H             ;Zeilennummer > End-Zeilennummer ?
2B53  C1                POP     BC              ;Programmzeiger in BC
2B54  DA 18 1A          JP      C,1A18H         ;ja, fertig
2B57  E3                EX      (SP),HL         ;Adresse nächste Zeile laden.
                                                ;End-Zeilennummer auf den Stack
2B58  E5                PUSH    HL              ;Adresse nächste Zeile auf Stack
2B59  C5                PUSH    BC              ;Programmzeiger auf den Stack
2B5A  EB                EX      DE,HL           ;Zeilennummer in HL
2B5B  22 EC 78          LD      (78ECH),HL      ;als '.'-Zeilennummer speichern
2B5E  CD AF 0F          CLL     0FAFH           ;Zeilennummer ausgeben
```

```
2B61  3E 20       LD    A,' '          ;Leerzeichen nach der Zeilennummer
2B63  E1          POP   HL             ;Programmzeiger laden
2B64  CD 2A 03    CALL  032AH          ;Leerzeichen ausgeben
2B67  CD 7E 2B    CALL  2B7EH          ;aus Zwischencode lesbaren Text erz
2B6A  2A A7 78    LD    HL,(78A7H)     ;Ein-/Ausgabe-Puffer adressieren
2B6D  CD 75 2B    CALL  2B75H          ;Text der Zeile ausgeben
2B70  CD FE 20    CALL  20FEH          ;Carriage-Return ausgeben
2B73  18 BE       JR    2B33H          ;nächste Zeile

                  ************************************

                  Text-String ausgeben (mit 00 abgeschlossen)
2B75  7E          LD    A,(HL)         ;Zeichen laden
2B76  B7          OR    A              ;=Textende ?
2B77  C8          RET   Z              ;ja, fertig!
2B78  CD 2A 03    CALL  032AH          ;Zeichen ausgeben
2B7B  23          INC   HL             ;Textadresse + 1
2B7C  18 F7       JR    2B75H          ;nächstes Zeichen

                  ************************************

                  Aus Zwischencode lesbaren Text erzeugen
                  Wird aus der Programmzeile in den Ein-/Ausgabe-Puffer
                  übertragen.
2B7E  E5          PUSH  HL             ;Programmzeiger auf den Stack
2B7F  2A A7 78    LD    HL,(78A7H)     ;Pufferzeiger in BC laden
2B82  44          LD    B,H
2B83  4D          LD    C,L
2B84  E1          POP   HL             ;Programmzeiger laden
2B85  16 FF       LD    D,0FFH         ;max.Zeichen = 255
2B87  18 03       JR    2B8CH
2B89  03          INC   BC             ;Pufferzeiger + 1
2B8A  15          DEC   D              ;Zeichenzähler - 1
2B8B  C8          RET   Z              ;Puffer voll? ja-fertig
2B8C  7E          LD    A,(HL)         ;Zeichen aus Programm laden
2B8D  B7          OR    A              ;Zeilenende ?
2B8E  23          INC   HL             ;Programmzeiger + 1
2B8F  02          LD    (BC),A         ;Zeichen in Puffer übertragen
2B90  C8          RET   Z              ;Zeilenende, fertig!
2B91  CD 9D 2E    JR    2E9DH          ;weiter bei 2E9DH (Rucksack)
2B94  FE FB       CP    0FBH           ;= ' - Token ?
2B96  20 08       JR    NZ,2BA0H       ;nein!
2B98  0B          DEC   BC             ;':REM' aus Puffer löschen
2B99  0B          DEC   BC             ;(Pufferzeiger - 4)
```

```
2B9A  0B          DEC   BC
2B9B  0B          DEC   BC
2B9C  14          INC   D             ;Zeichenzähler + 4
2B9D  14          INC   D
2B9E  14          INC   D
2B9F  14          INC   D
2BA0  FE 95       CP    95H           ;ELSE - Token ?
2BA2  CC 24 0B    CALL  Z,0B24H       ;ja, ':' davor entfernen
2BA5  D6 7F       SUB   7FH           ;Token - 7F = Nummer des Schl.worts
2BA7  E5          PUSH  HL            ;Programmzeiger auf den Stack
2BA8  5F          LD    E,A           ;Schlüsselwortnummer in E
2BA9  21 50 16    LD    HL,1650H      ;Schlüsselwort-Tabelle adressieren
2BAC  7E          LD    A,(HL)        ;Zeichen aus Tabelle laden
2BAD  B7          OR    A             ;Anfang eines neuen Schlüsselworts?
2BAE  23          INC   HL            ;Tabellenzeiger + 1
2BAF  F2 AC 2B    JP    P,2BACH       ;nein, weiter suchen
2BB2  1D          DEC   E             ;ja, gesuchtes Schlüsselwort ?
2BB3  20 F7       JR    NZ,2BACH      ;nein, nächstes Schlüsselwort such.
2BB5  E6 7F       AND   7FH           ;in 1.Zeichen Bit 7 löschen
2BB7  02          LD    (BC),A        ;Zeichen in Puffer übertragen
2BB8  03          INC   BC            ;Pufferzeiger + 1
2BB9  15          DEC   D             ;Zeichenzähler - 1
2BBA  CA D8 2B    JP    Z,2BD8H       ;Puffer voll, fertig
2BBD  7E          LD    A,(HL)        ;nächstes Schlüsselwort-Zeichen
2BBE  23          INC   HL            ;Tabelleneiger + 1
2BBF  B7          OR    A             ;neues Schlüsselwort ?
2BC0  F2 B7 2B    JP    P,2BB7H       ;nein, übertragen
2BC3  E1          POP   HL            ;Programmzeiger wieder laden
2BC4  18 C6       JR    2B8CH         ;nächstes Textzeichen

                  ++++++++++++++++++++++++++++++++++++

                  DELETE - Befehl
                  Programmzeilen löschen
2BC6  CD 10 1B    CALL  1B10H         ;beide Argumente analysieren
                                      ;1.Zeilenadr.in BC, 2.ZNr.auf Stack
2BC9  D1          POP   DE            ;2. Zeilennummer in DE
2BCA  C5          PUSH  BC            ;1. Zeilenadresse auf Stack
2BCB  C5          PUSH  BC            ;(2 mal)
2BCC  CD 2C 1B    CALL  1B2CH         ;2. Zeilenadresse ermitteln
2BCF  30 05       JR    NC,2BD6H      ;nicht vorh., FUNCTION CODE - Error
2BD1  54          LD    D,H           ;2. Zeilenadresse in DE
2BD2  5D          LD    E,L
2BD3  E3          EX    (SP),HL       ;1. Zeilenadresse laden
```

```
                                        ;2. Zeilenadresse auf den Stack
2BD4  E5          PUSH   HL            ;1. Zeilenadresse wieder auf Stack
2BD5  DF          RST    18H           ;1. Zeilenadresse <= 2. Zeilenadr.
2BD6  D2 4A 1E    JP     NC,1E4AH      ;nein, FUNCTION CODE - Error
2BD9  21 29 19    LD     HL,1929H      ;Text 'READY' adressieren
2BDC  CD A7 28    CALL   28A7H         ;und ausgeben
2BDF  C1          POP    BC            ;1. Zeilenadresse laden
2BE0  21 E8 1A    LD     HL,1AE8H      ;Rücksprungadresse laden
2BE3  E3          EX     (SP),HL       ;mit 2.Zeilenadr. tauschen
2BE4  EB          EX     DE,HL         ;2. Zeilenadresse in DE
2BE5  2A F9 78    LD     HL,(78F9H)    ;Programmtext-Ende in HL
2BE8  1A          LD     A,(DE)        ;Zeichen aus hint.Progbereich laden
2BE9  02          LD     (BC),A        ;und nach vorne übertragen
2BEA  03          INC    BC            ;Programmzeiger + 1
2BEB  13          INC    DE
2BEC  DF          RST    18H           ;Programmende erreicht ?
2BED  20 F9       JR     NZ,2BE8H      ;nein, nächstes Zeichen
2BEF  60          LD     H,B           ;letzte Zieladresse=neues Prog.ende
2BF0  69          LD     L,C
2BF1  22 F9 78    LD     (78F9H),HL    ;abspeichern
2BF4  C9          RET

                  ************************************

                  SOUND - Befehl

2BF5  CD 1C 2B    CALL   2B1CH         ;1.Parameter (Note) analysieren
2BF8  FE 20       CP     32            ;Note > 31 ?
2BFA  D2 4A 1E    JP     NC,1E4AH      ;ja, FUNCTION-CODE Error
2BFD  32 D2 7A    LD     (7AD2H),A     ;Notenwert speichern
2C00  CF          RST    8             ;folgt ein Komma ?
2C01  2C          DEFB   ','
2C02  CD 1C 2B    CALL   2B1CH         ;2.Parameter (Länge) analysieren
2C05  B7          OR     A             ;Tonlänge = 0 ?
2C06  CA 4A 1E    JP     Z,1E4AH       ;ja, FUNCTION CODE - Error
2C09  FE 0A       CP     10            ;Tonlänge > 9 ?
2C0B  D2 4A 1E    JP     NC,1E4AH      ;ja, FUNCTION CODE - Error
2C0E  F3          DI                   ;Interrupts ausschalten
2C0F  E5          PUSH   HL            ;Programmzeiger auf den Stack
2C10  3D          DEC    A             ;Tonlänge - 1
2C11  F5          PUSH   AF            ;auf den Stack
2C12  3A D2 7A    LD     A,(7AD2H)     ;Notenwert laden
2C15  B7          OR     A             ;= 0 ?
2C16  28 40       JR     Z,2C58H       ;ja, Pause
```

```
2C18  3D           DEC   A            ;Notenwert - 1
2C19  CB 27        SLA   A            ;* 2
2C1B  4F           LD    C,A          ;in BC = Offset für Frequenztabelle
2C1C  AF           XOR   A
2C1D  47           LD    B,A
2C1E  F1           POP   AF           ;Tonlänge laden
2C1F  21 CF 02     LD    HL,02CFH     ;Frequenztabelle adressieren
2C22  09           ADD   HL,BC        ;+ Offset für Note
2C23  5E           LD    E,(HL)       ;Frequenzwert aus Tabelle laden
2C24  23           INC   HL
2C25  56           LD    D,(HL)
2C26  D5           PUSH  DE           ;Frequenzwert auf den Stack
2C27  21 61 03     LD    HL,0361H     ;Tabelle der Zeitgrundwerte adress.
2C2A  CB 39        SRL   C            ;Frequenz-Offset / 2
2C2C  09           ADD   HL,BC        ;+Tabellen-Anfangsadresse
2C2D  5E           LD    E,(HL)       ;=Zeitgrundwert für 1/8 Note (in E)
2C2E  16 00        LD    D,0
2C30  21 21 03     LD    HL,0321H     ;Tabelle der Multiplikatoren für
                                      ;Tonlänge adressieren
2C33  4F           LD    C,A          ;Tonlänge-1 in BC (=Tab.Offset)
2C34  09           ADD   HL,BC        ;+Tabellen-Anfangsadresse
2C35  46           LD    B,(HL)       ;Multiplikator aus Tabelle laden
2C36  D5           PUSH  DE           ;Zeitgrundwert mit B+1 multipliz.
2C37  E1           POP   HL           ;Ergebnis in HL
2C38  19           ADD   HL,DE
2C39  10 FD        DJNZ  2C38H
2C3B  E5           PUSH  HL           ;Tondauer in BC übertragen
2C3C  C1           POP   BC           ;als Zyklus-Zähler
2C3D  E1           POP   HL           ;Frequenzwert laden
2C3E  CD F8 3A     CALL  3AFH         ;BREAK-Taste gedrückt ?
2C41  3A 3B 78     LD    A,(783BH)    ;Output-Latch - Byte laden
2C44  57           LD    D,A          ;in D
2C45  CD 69 34     CALL  3469H        ;Ton ausgeben
2C48  0B           DEC   BC           ;Zyklus-Zähler - 1
2C49  79           LD    A,C          ;= 0 ?
2C4A  B0           OR    B
2C4B  20 F1        JR    NZ,2C3EH     ;nein, Ton halten
2C4D  E1           POP   HL           ;Programmzeiger laden
2C4E  FB           EI                 ;Interrupts einschalten
2C4F  7E           LD    A,(HL)       ;Zeichen laden
2C50  23           INC   HL           ;Programmzeiger + 1
2C51  FE 3B        CP    ';'          ;folgt ein ';' ?
2C53  CA F5 2B     JP    Z,2BF5H      ;ja, nächste Note spielen
2C56  2B           DEC   HL           ;Programmzeiger - 1
```

```
2C57  C9              RET                 ;fertig

2C58  F1              POP   AF            ;Tonlänge - 1 laden
2C59  4F              LD    C,A           ;in BC
2C5A  AF              XOR   A
2C5B  47              LD    B,A
2C5C  21 21 03        LD    HL,0321H      ;Tabelle der Multiplikat.adressier.
2C5F  09              ADD   HL,BC         ;+ Tonlänge-1
2C60  46              LD    B,(HL)        ;Multiplikator aus Tabelle laden
2C61  21 36 19        LD    HL,6454H      ;Grundwert für 93.75 ms laden
2C64  E5              PUSH  HL            ;Grundwert mit B+1 multiplizieren
2C65  D1              POP   DE            ;= Pausenzähler
2C66  19              ADD   HL,DE
2C67  10 FD           DJNZ  2C66H
2C69  CD F8 3A        CALL  3AF8H         ;BREAK-Taste betätigt?
2C6C  2B              DEC   HL            ;Zähler - 1
2C6D  7D              LD    A,L           ;= 0 ?
2C6E  B4              OR    H
2C6F  20 F8           JR    NZ,2C69H      ;nein!
2C71  18 DA           JR    2C4DH         ;ja, Pause beendet!

                      ************************************

                      Ausgabe von Grafik-Zeichen auf einem Drucker
2C73  C5              PUSH  BC            ;BC auf Stack sichern
2C74  47              LD    B,A           ;Zeichen in B
2C75  3E 08           LD    A,8           ;Drucker in Grafik-Mode umschalten
2C77  CD BA 3A        CALL  3ABAH         ;durch Ausgabe von X'08'
2C7A  78              LD    A,B           ;Zeichern wieder in A
2C7B  E6 0F           AND   0FH           ;oberes Halbbyte löschen
2C7D  E5              PUSH  HL            ;HL auf Stack sichern
2C7E  CB 27           SLA   A             ;Zeichen * 2
2C80  4F              LD    C,A           ;als Tabellen-Offset in BC
2C81  AF              XOR   A
2C82  47              LD    B,A
2C83  21 AF 02        LD    HL,02AFH      ;Anfang Grafik-Tabelle adressieren
2C86  09              ADD   HL,BC         ;+ Zeichen-Offset
2C87  7E              LD    A,(HL)        ;1. Tabellenwert in B laden
2C88  47              LD    B,A
2C89  23              INC   HL            ;Tabellenadresse + 1
2C8A  7E              LD    A,(HL)        ;2. Tabellenwert in C laden
2C8B  4F              LD    C,A
2C8C  78              LD    A,B           ;1. Tabellenwert in A
2C8D  CD BA 3A        CALL  3ABAH         ;Wert dreimal auf den Drucker ausg.
```

```
2C90  CD BA 3A        CALL   3ABAH
2C93  CD BA 3A        CALL   3ABAH
2C96  79              LD     A,C             ;2. Tabellenwert in A
2C97  CD BA 3A        CALL   3ABAH           ;Wert dreimal auf den Drucker ausg.
2C9A  CD BA 3A        CALL   3ABAH
2C9D  CD BA 3A        CALL   3ABAH
2CA0  E1              POP    HL              ;HL und BC wiederherstellen
2CA1  C1              POP    BC
2CA2  3E 0F           LD     A,0FH           ;Drucker wieder in Text-Modus
2CA4  CD BA 3A        CALL   3ABAH           ;durch Ausgabe von X'0F'
2CA7  C9              RET                    ;fertig

                      ************************************

                      PEEK - Funktion
                      Inhalt einer Speicherstelle laden
2CAA  CD 7F 0A        CALL   0A7FH           ;Argument in Integer umwandeln (HL)
2CAD  7E              LDD    A,(HL)          ;Inhalt der Speicherstelle laden
2CAE  C3 F8 27        JP     27F8H           ;als Ergebnis in X

                      ************************************

                      POKE - Anweisung
                      Wert in Speicherstelle schreiben
2CB1  CD 02 2B        CALL   2B02H           ;Adresse auswerten und in DE
2CB4  D5              PUSH   DE              ;auf den Stack
2CB5  CF              RST    8               ;folgt ein Komma ?
2CB6  2C              DEFB   ','
2CB7  CD 1C 2B        CALL   2B1CH           ;Wert auswerten (<256) und in A
2CBA  D1              POP    DE              ;Adresse laden
2CBB  12              LD     (DE),A          ;Wert an dieser Adresse speichern
2CBC  C9              RET

                      ************************************

                      USING - Anweisung
                      Formatierte Ausgabe
2CBD  CD 38 23        CALL   2338H           ;Format-String auswerten
2CC0  CD F4 0A        CALL   0AF4H           ;kein String? TYPE MISMATCH - Error
2CC3  CF              RST    8               ;folgt ein Semikolon ?
2CC4  3B              DEFB   ';'
2CC5  EB              EX     DE,HL           ;Programmzeiger in DE
2CC6  2A 21 79        LD     HL,(7921H)      ;Zeiger auf Format-String laden
2CC9  18 08           JR     2CD3H
```

```
                        weitere Ausgaben mit gleichem Format-String
2CCB  3A DE 78          LD      A,(78DEH)       ;nächstes Zeichen laden
2CCE  B7                OR      A               ;= Anweisungsende?
2CCF  28 0C             JR      Z,2CDDH         ;ja, FUNCTION CODE - Error
2CD1  D1                POP     DE              ;Formatstring-Zeiger laden
2CD2  EB                EX      DE,HL           ;in HL
2CD3  E5                PUSH    HL              ;Formatstring-Zeiger auf Stack
2CD4  AF                XOR     A               ;letztes Zeichen löschen
2CD5  32 DE 78          LD      (78DEH),A
2CD8  BA                CP      D               ;Z-Flag löschen und Cy-Flag setzen.
2CD9  F5                PUSH    AF              ;Ergebnis-Flags auf den Stack
2CDA  D5                PUSH    DE              ;Programmzeiger auf den Stack
2CDB  46                LD      B,(HL)          ;Stringlänge laden
2CDC  B0                OR      B               ;= 0 ?
2CDD  CA 4A 1E          JP      Z,1E4AH         ;ja, FUNCTION CODE - Error
2CE0  23                INC     HL              ;Formatstring-Zeiger + 1
2CE1  4E                LD      C,(HL)          ;Stringadresse in HL
2CE2  23                INC     HL
2CE3  66                LD      H,(HL)
2CE4  69                LD      L,C
2CE5  18 1C             JR      2D03H

                        '%' - Feldlänge ermitteln
2CE7  58                LD      E,B             ;Stringlänge in E
2CE8  E5                PUSH    HL              ;Stringzeiger auf Stack
2CE9  0E 02             LD      C,2             ;Zeichenanzahl=2 (für Begrenzung)
2CEB  7E                LD      A,(HL)          ;Zeichen laden
2CEC  23                INC     HL              ;Stringzeiger + 1
2CED  FE 25             CP      '%'             ;= '%' ?
2CEF  CA 17 2E          JP      Z,2E17H         ;ja, formatierten String ausgeben
2CF2  FE 20             CP      20H             ;Leerzeichen ?
2CF4  20 03             JR      NZ,2CF9H        ;nein, Fehler!
2CF6  0C                INC     C               ;Zeichenzahl + 1
2CF7  10 F2             DJNZ    2CEBH           ;Stringlänge - 1 > 0 ? ja,zurück
2CF9  E1                POP     HL              ;Stringzeiger wieder laden
2CFA  43                LD      B,E             ;Stringlänge wieder in B
2CFB  3E 25             LD      A,'%'           ;'%' ausgeben

                        Anfang eines String- oder Nummernfeldes suchen
2CFD  CD 49 2E          CALL    2E49H           ;'+' außerhalb Nummernfeld ausg.
2D00  CD 2A 03          CALL    032AH           ;Zeichen ausgeben
2D03  AF                XOR     A               ;A = 0
2D04  5F                LD      E,A             ;Feldlänge = 0
```

```
2D05  57          LD    D,A         ;Format-Flag = 0
2D06  CD 49 2E    CALL  2E49H       ;'+' außerhalb Nummernfeld ausg.
2D09  57          LD    D,A         ;Format-Flag in D
2D0A  7E          LD    A,(HL       ;Zeichen aus String laden
2D0B  23          INC   HL          ;Stringzeiger + 1
2D0C  FE 21       CP    "!"         ;Ausrufezeichen ?
2D0E  CA 14 2E    JP    Z,2E14H     ;ja, 1. Zeichen des Strings drucken
2D11  FE 23       CP    "#"         ;Nummernzeichen ?
2D13  28 37       JR    Z,2D4CH     ;ja, Nummernfeld analysieren
2D15  05          DEC   B           ;Stringlänge - 1
2D16  CA FE 2D    JP    Z,2DFEH     ;= 0 ? ja, Stringende!
2D19  FE 2B       CP    '+'         ;= '+' ?
2D1B  3E 08       LD    A,8         ;Format-Flag = 8
2D1D  28 E7       JR    Z,2D06H     ;ja, Sprung
2D1F  2B          DEC   HL          ;Zeichen nochmals laden
2D20  7E          LD    A,(HL)
2D21  23          INC   HL
2D22  FE 2E       CP    '.'         ;Punkt ?
2D24  28 40       JR    Z,2D66H     ;ja, Nachkommastellen bestimmen
2D26  FE 25       CP    '%'         ;= '%' ?
2D28  28 BD       JR    Z,2CE7H     ;ja, String formatieren
2D2A  BE          CP    (HL)        ;= nächstes Zeichen ?
2D2B  20 D0       JR    NZ,2CFDH    ;nein, weiter
2D2D  FE 24       CP    '$'         ;2 Dollarzeichen ?
2D2F  28 14       JR    Z,2D45H     ;ja, Format-Flag setzen
2D31  FE 2A       CP    '*'         ;2 Sterne ?
2D33  20 C8       JR    NZ,2CFDH    ;nein, weiter
2D35  78          LD    A,B         ;letztes Zeichen noch im Formatstr.
2D36  FE 02       CP    2
2D38  23          INC   HL          ;Stringzeiger auf nächstes Zeichen
2D39  38 03       JR    C,2D3EH     ;nein!
2D3B  7E          LD    A,(HL)      ;Zeichen laden
2D3C  FE 24       CP    '$'         ;Dollarzeichen ?
2D3E  3E 20       LD    A,20H       ;Bit 5 des Format-Flags für '*' = 1
2D40  20 07       JR    NZ,2D49H    ;nein!
2D42  05          DEC   B           ;Stringlänge - 1
2D43  1C          INC   E           ;Nummernfeldlänge + 1
2D44  FE          DEFB  0EFH        ;CP 0AFH  Dummy-Befehl
2D45  AF          XOR   A           ;Format-Flag löschen
2D46  C6 10       ADD   A,10H       ;Bit 4 des Format-Flags für '$' = 1
2D48  23          INC   HL          ;Stringzeiger + 1
2D49  1C          INC   E           ;Nummernfeldlänge + 1
2D4A  82          ADD   A,D         ;Format-Flag mit letztem verknüpfen
2D4B  57          LD    D,A         ;und in D
```

```
2D4C  1C          INC   E             ;Nummernfeldlänge + 1
2D4D  0E 00       LD    C,0           ;Anzahl Nachkommastellen = 0
2D4F  05          DEC   B             ;Stringlänge - 1
2D50  28 47       JR    Z,2D99H       ;= 0? ja, Formatstring ausgewertet
2D52  7E          LD    A,(HLL)       ;Zeichen laden
2D53  23          INC   HL            ;Stringzeiger + 1
2D54  FE 2E       CP    '.'           ;Punkt ?
2D56  28 18       JR    Z,2D70H       ;ja, Anz.Nachkommastellen ermitteln
2D58  FE 23       CP    '#'           ;Nummernzeichen ?
2D5A  28 F0       JR    Z,2D4CH       ;ja, Nummernfeld weiter auswerten
2D5C  FE 2C       CP    ','           ;Komma ?
2D5E  20 1A       JR    NZ,2D7AH      ;nein, Nummenfeld-Parameter auswert
2D60  7A          LD    A,D           ;Bit 6 des Format-Flags für ',' = 1
2D61  F6 40       OR    40H
2D63  57          LD    D,A
2D64  18 E6       JR    2D4CH         ;weiter bei 2D4CH

                  Anzahl der Nachkommastellen bestimmen
2D66  7E          LD    A,(HL)        ;Zeichen laden
2D67  FE 23       CP    '#'           ;Nummernzeichen ?
2D69  3E 2E       LD    A,'.'
2D6B  20 90       JR    NZ,2CFDH      ;nein, '.' ausgeben
2D6D  0E 01       LD    C,1           ;Zähler f.Nachkommastellen = 1
2D6F  23          INC   HL            ;Stringzeiger + 1
2D70  0C          INC   C             ;Zähler f.Nachkommastellen + 1
2D71  05          DEC   B             ;Stringlänge - 1
2D72  28 25       JR    Z,2D99H       ;= 0 ? ja, Formatstringende!
2D74  7E          LD    A,(HL)        ;Zeichen laen
2D75  23          INC   HL            ;Stringzeiger + 1
2D76  FE 23       CP    '#'           ;Nummernzeichen ?
2D78  28 F6       JR    Z,2D70H       ;ja!

                  Nummernfeld-Endparameter auswerten
2D7A  D5          PUSH  DE            ;Format-Flag auf den Stack
2D7B  11 97 2D    LD    DE,2D97H      ;Rücksprungadresse setzen
2D7E  D5          PUSH  DE
2D7F  54          LD    D,H           ;Stringzeiger in DE
2D80  5D          LD    E,L
2D81  FE 5B       CP    5BH           ;letzt.Zeichen = 'Pfeil hoch' ?
2D83  C0          RET   NZ            ;nein, weiter bei 2D97H
2D84  BE          CP    (HL)          ;auch die nächsten 3 Zeichen ?
2D85  C0          RET   NZ            ;nein!
2D86  23          INC   HL
2D87  BE          CP    (HL)
```

```
2D88  C0          RET   NZ          ;nein!
2D89  23          INC   HL
2D8A  BE          CP    (HL)
2D8B  C0          RET   NZ          ;nein!
2D8C  23          INC   HL
2D8D  78          LD    A,B         ;Stringlänge < 4 ?
2D8E  D6 04       SUB   4
2D90  D8          RET   C           ;ja, die 4 Pfeile ignorieren
2D91  D1          POP   DE          ;Rücksprungadresse vom Stack
2D92  D1          POP   DE          ;Format-Flag laden
2D93  47          LD    B,A         ;Stringlänge - 4
2D94  14          INC   D           ;Bit 1 des Format-Flags für
                                    ;Exponentenausgabe setzen
2D95  23          INC   HL          ;Stringzeiger + 1
2D96  CA          DEFB  0CAH        ;JP Z,D1EB  Dummy-Befehl
2D97  EB          EX    DE,HL       ;Stringzeiger wieder in HL
2D98  D1          POP   DE          ;Format-Flag laden
2D99  7A          LD    A,D         ;Format-Flag in A
2D9A  2B          DEC   HL          ;Stringzeiger - 1
2D9B  1C          INC   E           ;Nummernfeldlänge + 1
2D9C  E6 08       AND   8           ;'+' - Bit gesetzt ?
2D9E  20 15       JR    NZ,2DB5H    ;ja!
2DA0  1D          DEC   E           ;Nummernfeldlänge - 1
2DA1  78          LD    A,B         ;Stringlänge = 0 ?
2DA2  B7          OR    A
2DA3  28 10       JR    Z,2DB5H     ;ja, Formatstring ausgewertet
2DA5  7E          LD    A,(HL)      ;Zeichen laden
2DA6  D6 2D       SUB   '-'         ;Minuszeichen ?
2DA8  28 06       JR    Z,2DB0H     ;ja!
2DAA  FE FE       CP    0FEH        ;Pluszeichen ?
2DAC  20 07       JR    NZ,2DB5H    ;nein, Ausgabe!
2DAE  3E 08       LD    A,8         ;Bit 3 des Format-Flags für '+' = 1
2DB0  C6 04       ADD   A,4         ;Bit 2 des Format-Flags für
                                    ;Vorzeichen hinter der Zahl = 1
2DB2  82          ADD   A,D         ;zu Gesamt-Formatflag verknüpfen
2DB3  57          LD    D,A         ;und in D
2DB4  05          DEC   B           ;Stringlänge - 1
2DB5  E1          POP   HL          ;Programmzeiger laden
2DB6  F1          POP   AF          ;Flags wieder laden
2DB7  28 50       JR    Z,2E09H     ;Anweisungsende ? ja, fertig!
2DB9  C5          PUSH  BC          ;Stringlänge + Nachkommastellen
                                    ;auf den Stack
2DBA  D5          PUSH  DE          ;Format-Flag + Nummernfeldlänge
                                    ;auf den Stack
```

```
2DBB  CD 37 23     CALL  2337H        ;Ausdruck auswerten (zu form.Zahl)
2DBE  D1           POP   DE           ;Flag+Nummernfeldlänge laden
2DBF  C1           POP   BC           ;Stringlng+Nachkommast. laden
2DC0  C5           PUSH  BC           ;und wieder auf den Stack
2DC1  E5           PUSH  HL           ;Programmzeiger auf den Stack
2DC2  43           LD    B,E          ;Nummernfeldlänge in B
2DC3  78           LD    A,B          ;+ Nachkommastellen
2DC4  81           ADD   A,C
2DC5  FE 19        CP    19H          ;Gesamtfeldlänge >= 25 ?
2DC7  D2 4A 1E     JP    NC,1E4AH     ;ja, FUNCTION CODE - Error
2DCA  7A           LD    A,D          ;Format-Flag in A
2DCB  F6 80        OR    80H          ;Bit 7 setzen (Formatierung!)
2DCD  CD BE 0F     CALL  0FBEH        ;Zahl in formatierten String umw.
2DD0  CD A7 28     CALL  28A7H        ;und diesen ausgeben
2DD3  E1           POP   HL           ;Programmzeiger laden
2DD4  2B           DEC   HL           ;Programmzeiger - 1
2DD5  D7           RST   10H          ;nächstes Zeichen laden
2DD6  37           SCF                ;Carry setzen (für CR)
2DD7  28 0D        JR    Z,2DE6H      ;Anweisungsende ? ja, Sprung
2DD9  32 DE 78     LD    (78DEH),A    ;Zeichen speichern
2DDC  FE 3B        CP    ';'          ;Semikolon ?
2DDE  28 05        JR    Z,2DE5H      ;ja!
2DE0  FE 2C        CP    ','          ;Komma ?
2DE2  C2 97 19     JP    NZ,1997H     ;nein, SYNTAX ERROR
2DE5  D7           RST   10H          ;nächstes Zeichen
2DE6  C1           POP   BC           ;Stringzähler laden
2DE7  EB           EX    DE,HL        ;Programmzeiger in DE
2DE8  E1           POP   HL           ;Stringzeiger laden
2DE9  E5           PUSH  HL           ;und wieder auf den Stack
2DEA  F5           PUSH  AF           ;Flags auf den Stack
2DEB  D5           PUSH  DE           ;Programmzeiger auf den Stack
2DEC  7E           LD    A,(HL)       ;Ursprüngliche Stringlänge in A
2DED  90           SUB   B            ;-Stringlänge =Anz.verarb. Zeichen
2DEE  23           INC   HL           ;Stringzeiger + 1
2DEF  4E           LD    C,(HL)       ;Stringadresse in HL laden
2DF0  23           INC   HL
2DF1  66           LD    H,(HL)
2DF2  69           LD    L,C
2DF3  16 00        LD    D,0          ;Anzahl der verarbeiteten Zeichen
2DF5  5F           LD    E,A          ;in DE
2DF6  19           ADD   HL,DE        ;+Stringadresse = Adresse des
                                      ;Reststrings
2DF7  78           LD    A,B          ;Reststringlänge > 0 ?
2DF8  B7           OR    A
```

```
2DF9  C2 03 2D        JP      NZ,2D03H        ;ja, weiter
2DFC  18 06           JR      2E04H

                      Formatstring-Ende
2DFE  CD 49 2E        CALL    2E49H           ;'+' außerhalb Nummernfeld ausgeben
2E01  CD 2A 03        CALL    032AH           ;Zeichen ausgeben
2E04  E1              POP     HL              ;Programmzeiger laden
2E05  F1              POP     AF              ;Flag wieder laden. Anweisungsende?
2E06  C2 CB 2C        JP      NZ,2CCBH        ;nein, nächstes Zahl mit gleichem
                                              ;Formatstring formatieren
2E09  DC FE 20        CALL    C,20FEH         ;Carry gesetzt? ja, CR ausgeben
2E0C  E3              EX      (SP),HL         ;Programmzeiger auf den Stack
                                              ;Stringzeiger laden
2E0D  CD DD 29        CALL    29DDH           ;Formatstring aus Stringbereich
                                              ;und Zwischenspeicher löschen
2E10  E1              POP     HL              ;Programmzeiger laden
2E11  C3 69 21        JP      2169H           ;Ausgabe-Flag auf Bildschirm
                                              ;und fertig!

                      Stringformatierung
2E14  0E 01           LD      C,1             ;Ansprung '!', Zeichenzahl = 1
2E16  3E              DEFB    3EH             ;LD A,F1  Dummy-Befehl
2E17  F1              POP     AF              ;Ansprung '%', Stackkorrektur
2E18  05              DEC     B               ;Stringlänge - 1
2E19  CD 49 2E        CALL    2E49H           ;'+' außerhalb Nummernfeld ausgeben
2E1C  E1              POP     HL              ;Programmzeiger laden
2E1D  F1              POP     AF              ;Flag wieder laden
2E1E  29 E9           JR      Z,2E09H         ;Anweisungsende? ja, fertig
2E20  C5              PUSH    BC              ;Stringlänge auf den Stack
2E21  CD 37 23        CALL    2337H           ;Ausdruck auswerten
                                              ;(zu formatierender String)
2E24  CD F4 0A        CALL    0AF4H           ;Ergeb. kein String? TYPE MISMATCH
2E27  C1              POP     BC              ;Stringlänge laden
2E28  C5              PUSH    BC              ;und wieder auf den Stack
2E29  E5              PUSH    HL              ;Programmzeiger auf den Stack
2E2A  2A 21 79        LD      HL,(7921H)      ;Stringzeiger des zu formatierenden
                                              ;Strings laden
2E2D  41              LD      B,C             ;Zeichenanzahl als 2. Argument
                                              ;für LEFT$ in B
2E2E  0E 00           LD      C,0             ;Left-Offset = 0
2E30  C5              PUSH    BC              ;beide Parameter auf den Stack
2E31  CD 68 2A        CALL    2A68H           ;String formatieren. Das 1. '!'
                                              ;o. die ersten '%' abtrennen
2E34  CD AA 28        CALL    28AAH           ;Formatierten String ausgeben
```

```
2E37  2A 21 79        LD    HL,(7921H)     ;Stringzeiger in HL
2E3A  F1              POP   AF             ;Zeichenanzahl in A
2E3B  96              SUB   (HL)           ;Länge des formatierten Strings
2E3C  47              LD    B,A            ;- Zeichenzahl in B
2E3D  3E 20           LD    A,' '          ;Leerzeichen laden
2E3F  04              INC   B              ;Differenz = 0 ?
2E40  05              DEC   B
2E41  CA D3 2D        JP    Z,2DD3H        ;ja, weiter
2E44  CD 2A 03        CALL  032AH          ;Leerzeichen ausgeben
2E47  18 F7           JR    2E40H          ;weiter bei 2E40H

                      Unterprogramm für USING
2E49  F5              PUSH  AF             ;AF sichern
2E4A  7A              LD    A,D            ;Bit im Format-Flag gesetzt ?
2E4B  B7              OR    A              ;(kann nur '+'-Bit sein)
2E4C  3E 2B           LD    A,'+'          ;'+' laden
2E4E  C4 2A 03        CALL  NZ,032AH       ;ja, ausgeben
2E51  F1              POP   AF             ;AF wieder laden
2E52  C9              RET

                      ************************************

                      Ausgabe einer vorhandenen Zeile bei AUTO-Eingabe
2E53  60              LD    H,B            ;Zeilenadresse in HL
2E54  69              LD    L,C
2E55  23              INC   HL             ;Zeilenadresse hinter Zeilennummer
2E56  23              INC   HL
2E57  23              INC   HL
2E58  23              INC   HL
2E59  CD 7E 2B        CALL  2B7EH          ;Zeile aufbereiten u. in I/O-Buffer
2E5C  2A A7 78        LD    HL,(78A7H)     ;Bufferadresse laden
2E5F  CD 75 2B        CALL  2B75H          ;Zeile ausgeben
2E62  C9              RET

                      ************************************

                      MODE - Anweisung ausführen
2E63  CF              RST   8              ;folgt ein '(' ?
2E64  28              DEFB  '('
2E65  CD 1C 2B        CALL  2B1CH          ;Operand in Klammer auswerten
2E68  B7              OR    A              ;MODE (0) ?
2E69  28 12           JR    Z,2E7DH        ;ja, Sprung
2E6B  3D              DEC   A              ;- 1
2E6C  28 03           JR    Z,2E71H        ;= 0? ja, MODE (1)!
```

```
2E6E  C3 4A 1E      JP    1E4AH          ;nein, FUNCTION CODE - Error

                    MODE (1) setzen
2E71  16 00         LD    D,0            ;Löschzeichen = X'00'
2E73  3A 3B 78      LD    A,(783BH)      ;Output-Latch Byte laden
2E76  F6 08         OR    8              ;Bit 3 setzen
2E78  32 3B 78      LD    (783BH),A      ;und zurückspeichern
2E7B  18 0A         JR    2E87H          ;weiter bei 2E87H

                    MODE (0) setzen
2E7D  16 20         LD    D,' '          ;Löschzeichen = Blank
2E7F  3A 3B 78      LD    A,(783BH)      ;Output-Latch Byte laden
2E82  E6 F7         AND   0F7H           ;Bit 3 löschen
2E84  32 3B 78      LD    (783BH),A      ;und zurückspeichern

2E87  32 00 68      LD    (6800H),A      ;Latch-Byte ausgeben
2E8A  E5            PUSH  HL             ;Programmzeiger auf Stack
2E8B  21 00 70      LD    HL,7000H       ;Bildanfangsadresse laden
2E8E  01 00 08      LD    BC,2048        ;Zähler = 2K-Byte
2E91  7A            LD    A,D            ;Bildspeicher mit Löschzeichen
2E92  77            LD    (HL),A         ;füllen
2E93  23            INC   HL             ;Bildadresse + 1
2E94  0B            DEC   BC             ;Zähler - 1
2E95  78            LD    A,B            ;= 0 ?
2E96  B1            OR    C
2E97  20 F8         JR    NZ,2E91H       ;nein, weiter
2E99  E1            POP   HL             ;Programmzeiger laden
2E9A  CF            RST   8              ;Abschluß mit ')' ?
2E9B  29            DEFB  ')'
2E9C  C9            RET

                    ************************************

                    Zusatzroutine für LIST
                    (Ausgabe von Strings)
2E9D  FE 22         CP    '"'            ;Stringanfang ?
2E9F  CA B3 2E      JP    Z,2EB3H        ;ja, weiter bei 2EB3H
2EA2  B7            OR    A              ;ist es ein Token ?
2EA3  F2 B9 2B      JP    P,2B89H        ;nein, nächstes Zeichen
2EA6  C3 94 2B      JP    2B94H          ;ja, weiter bei 2B94H

2EA9  7E            LD    A,(HL)         ;Zeichen laden
2EAA  B7            OR    A              ;Test auf Zeilenende
2EAB  23            INC   HL             ;Programmzeiger + 1
```

```
2EAC  02          LD    (BC),A        ;Zeichen in Puffer übertragen
2EAD  C8          RET   Z             ;Zeilenende, fertig
2EAE  FE 22       CP    '"'           ;Stringende ?
2EB0  CA 89 2B    JP    Z,2B89H       ;ja, weiter bei 2B89H
2EB3  03          INC   BC            ;Pufferzeiger + 1
2EB4  15          DEC   D             ;Puffer voll ?
2EB5  C8          RET   Z             ;ja, aufhören
2EB6  18 F1       JR    2EA9H         ;nächstes Zeichen im String

                  ************************************

                  Interrupt - Service - Routine
                  (wird alle 20 ms vom Video-Controller angestoßen)
2EB8  F5          PUSH  AF            ;Registerinhalte retten
2EB9  C5          PUSH  BC
2EBA  D5          PUSH  DE
2EBB  E5          PUSH  HL
2EBC  CD 7D 78    CALL  787DH         ;RAM - Erweiterungsausgang
2EBF  CD 7B 3F    CALL  3F7BH         ;Bildschirm ggf. invertieren
                                      ;Print-Buffer ausgeben
2EC2  CD DC 2E    CALL  2EDCH         ;Cursor ausgeben/blinken
2EC5  CD FD 2E    CALL  2EFDH         ;Tastatur abfragen
2EC8  F5          PUSH  AF            ;eingelesenes Zeichen sichern
2EC9  21 39 78    LD    HL,7839H      ;Flag 2 adressieren
2ECC  CB 46       BIT   0,(HL)        ;Carriage-Return Flag gesetzt?
2ECE  CC 1B 30    CALL  Z,301BH       ;nein, Zeichen ausgeben (Echo)
2ED1  F1          POP   AF            ;Zeichen wieder laden
2ED2  CD 30 34    CALL  3430H         ;Summer ertönen lassen
2ED5  E1          POP   HL            ;Registerinhalte wiederherstellen
2ED6  D1          POP   DE
2ED7  C1          POP   BC
2ED8  F1          POP   AF
2ED9  FB          EI                  ;Interupts wieder einschalten
2EDA  ED 4D       RETI                ;RETURN vom Interrupt

                  ************************************

                  Cursor ausgeben / blinken
2EDC  3A 39 78    LD    A,(7839H)     ;Flag 2 laden
2EDF  CB 47       BIT   0,A           ;Carriage-Return Flag gesetzt?
2EE1  C0          RET   NZ            ;ja, fertig
2EE2  21 41 78    LD    HL,7841H      ;Blinkzähler adressieren
2EE5  35          DEC   (HL)          ;- 1
2EE6  C0          RET   NZ            ;nicht Null, fertig!
```

```
2EE7  3E 10       LD    A,16          ;Blinkzähler neu setzen
2EE9  32 41 78    LD    (7841H),A
2EEC  2A 20 78    LD    HL,(7820H)    ;Cursor-Adresse laden
2EEF  3E 40       LD    A,40H         ;Inverse-Bit in A setzen
2EF1  AE          XOR   (HL)          ;Zeichen an Cursor-Pos. invertieren
2EF2  77          LD    (HL),A
2EF3  C9          RET

                  ************************************

                  Ein Zeichen von der Tastatur einlesen
                  (Aufruf über Tastatur-DCB)
2EF4  CD FD 2E    CALL  2EFDH         ;Tastatur auswerten
2EF7  F5          PUSH  AF            ;Zeichen sichern
2EF8  CD 0E 2F    CALL  2F0EH         ;Flags zurücksetzen
2EFB  F1          POP   AF            ;Zeichen wieder laden
2EFC  C9          RET

                  ************************************

                  Tastatur einmal auswerten
                  Ausg.: A = eingelesenes ASCII-Zeichen
                         A = 0, wenn keine Taste betätigt ist
2EFD  3A 00 68    LD    A,(6800H)     ;alle Zeilen der Tastaturmatrix les
2F00  F6 C0       OR    11000000B     ;Spalten 6 u. 7 ausblenden
2F02  2F          CPL                 ;A-Register invertieren
2F03  FE 00       CP    0             ;wenn = 0, keine Taste gedrückt
2F05  28 07       JR    Z,2F0EH       ;dann alle Flags löschen
2F07  CD 28 2F    CALL  2F28H         ;Matrixzeilen einzeln auswerten
2F0A  B7          OR    A             ;kein ASCII - Zeichen ?
2F0B  C2 D7 05    JP    NZ,05D7H      ;doch, prüfen auf Mehrfachbetätig.

                  ************************************

                  Flag-Bits zurücksetzen
2F0E  21 38 78    LD    HL,7838H      ;Flag 1 adressieren
2F11  CB 56       BIT   2,(HL)        ;Funktions-Flag gesetzt?
2F13  28 08       JR    Z,2F1DH       ;nein!
2F15  3A 3A 78    LD    A,(783AH)     ;Zeitzähler laden
2F18  B7          OR    A             ;= 0 ?
2F19  28 02       JR    Z,2F1DH       ;ja, Funktions-Flag nicht löschen
2F1B  CB 96       RES   2,(HL)        ;Funktions-Flag löschen
2F1D  7E          LD    A,(HL)        ;Flag 1 laden
2F1E  E6 06       AND   00000110B     ;bis auf Funkt.+ Inv.-Flag löschen
```

```
2F20  32 38 78        LD    (7838H),A      ;und zurückspeichern
2F23  AF              XOR   A              ;Zeichenpuffer B1 löschen
2F24  32 36 78        LD    (7836H),A
2F27  C9              RET

                      ************************************

                      Tastatur zeilenweise auswerten
                      Ausg.: A = Tastencode oder Null
2F28  21 FE 68        LD    HL,68FEH       ;Tastaturzeile 1 adressieren
2F2B  0E 08           LD    C,8            ;Zeilenzähler = 8
2F2D  06 06           LD    B,6            ;Spaltenzähler = 6
2F2F  7E              LD    A,(HL)         ;Zeileninhalt laden
2F30  F6 04           OR    00000100B      ;Spalte 2 (Sonderfunkt.) ausblenden
2F32  1F              RRA                  ;niederwertigstes Bit in Carry
2F33  30 2D           JR    NC,2F62H       ;= 0, Taste gefunden
2F35  10 FB           DJNZ  2F32H          ;nächstes Bit (8x)
2F37  CB 05           RLC   L              ;nächste Zeile adressieren
2F39  0D              DEC   C              ;Zeilenzähler - 1
2F3A  20 F1           JR    NZ,2F2DH       ;> 0 ? ja, nächste Zeile
2F3C  06 04           LD    B,4            ;Spaltenzähler = 4 (Spalte 2)
2F3E  21 DF 68        LD    HL,68DFH       ;Tastaturzeile 6 adressieren
2F41  7E              LD    A,(HL)         ;Zeileninhalt laden
2F42  CB 57           BIT   2,A            ;'-' Taste gedrückt ?
2F44  28 10           JR    Z,2F56H        ;ja!
2F46  CB 05           RLC   L              ;Tastaturzeile 7 adressieren
2F48  7E              LD    A,(HL)         ;Zeileninhalt laden
2F49  CB 57           BIT   2,A            ;RETURN - Taste ?
2F4B  28 0D           JR    Z,2F5AH        ;ja!
2F4D  CB 05           RLC   L              ;Tastaturzeile 8 adressieren
2F4F  7E              LD    A,(HL)         ;Zeileninhalt laden
2F50  CB 57           BIT   2,A            ;':' Taste gedrückt ?
2F52  28 0A           JR    Z,2F5EH        ;ja!
2F54  AF              XOR   A              ;mit A = 0 zurück
2F55  C9              RET

2F56  0E 03           LD    C,3            ;Zeilenzähler = 3 (für '-')
2F58  18 06           JR    2F60H
2F5A  0E 02           LD    C,2            ;Zeilenzähler = 2 (für RETURN)
2F5C  18 02           JR    2F60H
2F5E  0E 01           LD    C,1            ;Zeilenzähler = 1 (für ':')
2F60  F6 04           OR    00000100B      ;Spalte 2 wieder ausblenden

2F62  5F              LD    E,A            ;Zeileninhalt in E merken !!!
```

```
2F63  3E 06        LD    A,6          ;Aus Spalten- und Zeilenzähler
2F65  90           SUB   B            ;Offset für die Tastatur-tabellen
2F66  CB 27        SLA   A            ;ermitteln.
2F68  CB 27        SLA   A            ;
2F6A  CB 27        SLA   A            ;A = 8 ( 6 - B ) + 8 - C
2F6C  C6 08        ADD   A,8
2F6E  91           SUB   C

2F6F  ED 43 42 78  LD    (7842H),BC   ;Zeilen- und Spaltenzähler merken
2F73  22 44 78     LD    (7844H),HL   ;Zeilenadresse merken

2F76  21 D9 01     LD    HL,01D9H     ;Tastaturtabelle 1 (o. SHIFT) adr.
2F79  4F           LD    C,A          ;Tabellen-Offset in BC
2F7A  06 00        LD    B,0
2F7C  3A FB 68     LD    A,(68FBH)    ;Tastaturzeile 3 laden
2F7F  CB 57        BIT   2,A          ;SHIFT - Taste gedrückt ?
2F81  20 0A        JR    NZ,2F8DH     ;nein!
2F83  21 38 78     LD    HL,7838H     ;Flag 1 adressieren
2F86  CB C6        SET   0,(HL)       ;SHIFT-Flag setzen
2F88  21 09 02     LD    HL,0209H     ;Tastaturtabelle 2 (m. SHIFT) adr.
2F8B  18 3D        JR    2FCAH        ;Code aus Tabelle lesen
2F8D  3A FD 68     LD    A,(68FDH)    ;Tastaturzeile 2 laden
2F90  CB 57        BIT   2,A          ;CTRL - Taste gedrückt ?
2F92  20 39        JR    NZ,2FCDH     ;nein!
2F94  3A 7F 68     LD    A,(687FH)    ;Tastaturzeile 8 laden
2F97  CB 57        BIT   2,A          ;':' - Taste gedrückt ? (INVERSE)
2F99  20 0E        JR    NZ,2FA9H     ;nein!
2F9B  21 38 78     LD    HL,7838H     ;Flag 1 adressieren
2F9E  CB 6E        BIT   5,(HL)       ;WARTE-Flag gesetzt ?
2FA0  20 04        JR    NZ,2FA6H     ;ja, Tastendruck ignorieren
2FA2  7E           LD    A,(HL)       ;Flag 1 laden
2FA3  EE 22        XOR   00100010B    ;INVERSE-Flag invertieren,
                                      ;WARTE-Flag setzen
2FA5  77           LD    (HL),A       ;Flag 1 speichern
2FA6  AF           XOR   A            ;A = 0
2FA7  C1           POP   BC           ;1. Rücksprungadr. entfernen
2FA8  C9           RET                ;zwei Ebenen zurück
2FA9  21 38 78     LD    HL,7838H     ;Flag 1 adressieren
2FAC  CB FE        SET   7,(HL)       ;CONTROL-Flag setzen
2FAE  CB 56        BIT   2,(HL)       ;FUNCTION-Flag gesetzt ?
2FB0  28 05        JR    Z,2FB7H      ;nein!
2FB2  21 69 02     LD    HL,0269H     ;Tastaturtabelle 4 (Funktionen)
2FB5  18 13        JR    2FCAH        ;Code aus Tabelle lesen
2FB7  3A BF 68     LD    A,(68BFH)    ;Tastaturzeile 7 laden
```

```
2FBA  CB 57        BIT   2,A          ;RETURN gedrückt ?
2FBC  20 07        JR    NZ,2FC5H     ;nein!
2FBE  CB D6        SET   2,(HL)       ;FUNCTION-Flag setzen
2FC0  AF           XOR   A            ;Zeitzähler rücksetzen
2FC1  32 3A 78     LD    (783AH),A
2FC4  C9           RET

2FC5  CB 96        RES   2,(HL)       ;FUNCTION-Flag löschen
2FC7  21 39 02     LD    HL,0239H     ;Tastaturtabelle 3 (Keywords)
2FCA  09           ADD   HL,BC        ;Tabellenadr. + Offset
2FCB  7E           LD    A,(HL)       ;Code aus Tabelle lesen
2FCC  C9           RET                ;zurück

2FCD  3A 38 78     LD    A,(7838H)    ;Flag 1 laden
2FD0  E6 81        AND   10000001B    ;SHIFT- o. CTRL-Flag gesetzt ?
2FD2  28 F6        JR    Z,2FCAH      ;nein, Code ermitteln
2FD4  AF           XOR   A            ;Taste ignorieren
2FD5  E1           POP   HL           ;1. Rücksprungadr. entfernen
2FD6  C9           RET                ;zwei Ebenen zurück

                   ************************************

                   Tastenwiederholung
2FD7  21 38 78     LD    HL,7838H     ;Flag 1 adressieren
2FDA  CB 6E        BIT   5,(HL)       ;WARTE-Flag gesetzt ?
2FDC  28 25        JR    Z,3003H      ;nein!
2FDE  3A 3A 78     LD    A,(783AH)    ;Zeitzähler + 1
2FE1  3C           INC   A
2FE2  32 3A 78     LD    (783AH),A
2FE5  FE 2A        CP    42           ;= 0.84 Sekunden ?
2FE7  28 02        JR    Z,2FEBH      ;ja!
2FE9  AF           XOR   A            ;zurück mit A = 0
2FEA  C9           RET

2FEB  7E           LD    A,(HL)       ;Flag 1 laden
2FEC  E6 DF        AND   11011111B    ;WARTE-Flag löschen
2FEE  F6 40        OR    01000000B    ;WIEDERHOL-Flag setzen
2FF0  32 38 78     LD    (7838H),A    ;Flag 1 speichern
2FF3  AF           XOR   A            ;Zeitzähler zurücksetzen
2FF4  32 3A 78     LD    (783AH),A
2FF7  CB 66        BIT   4,(HL)       ;2 Tasten gedrückt ?
2FF9  20 04        JR    NZ,2FFFH     ;ja!
2FFB  3A 36 78     LD    A,(7836H)    ;Tastencode aus Zeichenpuffer laden
2FFE  C9           RET                ;zurück
```

```
2FFF  3A 37 78        LD    A,(7837H)      ;Code für 2. Taste laden
3002  C9              RET                  ;zurück
3003  CB 76           BIT   6,(HL)         ;WIEDERHOL-Flag gesetzt ?
3005  20 07           JR    NZ,300EH       ;ja!
3007  CB EE           SET   5,(HL)         ;WARTE-Flag setzen
3009  AF              XOR   A              ;Zeitzähler zurücksetzen
300A  32 3A 78        LD    (783AH),A
300D  C9              RET                  ;mit A = 0 zurück
300E  3A 3A 78        LD    A,(783AH)      ;Zeitzähler + 1
3011  3C              INC   A
3012  32 3A 78        LD    (783AH),A
3015  FE 06           CP    6              ;= 0.12 Sekunden ?
3017  28 DA           JR    Z,2FF3H        ;ja!
3019  AF              XOR   A              ;mit A = 0
301A  C9              RET                  ;zurück

                      ************************************

                      Eingegebenes Zeichen auf dem Bildschirm darstellen
                      (ECHO - Funktion)
301B  B7              OR    A              ;A = 0 ? (kein Zeichen)
301C  C8              RET   Z              ;ja, fertig
301D  F5              PUSH  AF             ;Zeichen sichern
301E  CD 39 30        CALL  3039H          ;Zeichen ausgeben
3021  F1              POP   AF             ;Zeichen wieder laden
3022  FE 0D           CP    0DH            ;war es ein Carriage-Return?
3024  C8              RET   Z              ;ja, fertig
3025  FE 01           CP    1              ;oder BREAK ?
3027  C8              RET   Z              ;ja, auch fertig
3028  3A 39 78        LD    A,(7839H)      ;Flag 2 laden
302B  CB 47           BIT   0,A            ;CR-Flag gesetzt ?
302D  C0              RET   NZ             ;ja, fertig
302E  3E 20           LD    A,32           ;Blinkzähler auf dopp. Wert (Pause)
3030  32 41 78        LD    (7841H),A
3033  2A 20 78        LD    HL,(7820H)     ;Cursor-Adresse laden
3036  C3 B2 3E        JP    3EB2H          ;Zeichen invertiert ausgeben

                      ************************************

                      Direktausgabe eines Zeichens oder Schlüsselworts
3039  21 38 78        LD    HL,7838H       ;Flag 1 adressieren
303C  CB 7E           BIT   7,(HL)         ;CONTROL-Flag gesetzt?
303E  CA 57 31        JP    Z,3157H        ;nein, Zeichen ausgeben
3041  B7              OR    A              ;Ist es ein Token?
```

```
3042  F2 57 31        JP      P,3157H         ;nein, Zeichen ausgeben

3045  F5              PUSH    AF              ;Zeichen sichern
3046  D6 80           SUB     80H             ;Bit 7 löschen
3048  3C              INC     A               ;+ 1 als Wort-Zähler
3049  47              LD      B,A
304A  21 4F 16        LD      HL,164FH        ;Adresse der Schlüsselwort-Tab. -1
304D  23              INC     HL              ;Adresszeiger + 1
304E  CB 7E           BIT     7,(HL)          ;Beginn eines neuen Wortes ?
3050  28 FB           JR      Z,304DH         ;nein, weiter
3052  10 F9           DJNZ    304DH           ;Wortzähler - 1 = 0 ?
3054  7E              LD      A,(HL)          ;ja, gesuchtes Schlüsselwort
3055  CD 82 30        CALL    3082H           ;Zeichen aus Tabelle ausgeben
3058  7E              LD      A,(HL)          ;nächstes Zeichen
3059  CB 7F           BIT     7,A             ;neues Schlüsselwort ?
305B  28 F8           JR      Z,3055H         ;nein, weiter ausgeben
305D  F1              POP     AF              ;Zeichen aus Stack wieder laden
                      Prüfen, ob dem Token das Zeichen '(' angefügt wird.
305E  06 16           LD      B,22            ;Anzahl der Tabellenelemente
3060  21 99 02        LD      HL,0299H        ;Anfangsadresse der Tabelle
3063  BE              CP      (HL)            ;Token = Tabelleneintrag ?
3064  28 16           JR      Z,307CH         ;ja, Sprung
3066  23              INC     HL              ;Adresszeiger + 1
3067  10 FA           DJNZ    3063H           ;nächsten Tabelleneintrag
                      nicht in Tabelle, Sonderbehandlung 'DEF'
3069  FE B0           CP      0B0H            ;DEF-Token ?
306B  C0              RET     NZ              ;nein, fertig!
306C  3E 20           LD      A,' '           ;ja, zu 'DEF FN' ergänzen
306E  CD 82 30        CALL    3082H
3071  3E 46           LD      A,'F'
3073  CD 82 30        CALL    3082H
3076  3E 4E           LD      A,'N'
3078  CD 82 30        CALL    3082H
307B  C9              RET                     ;fertig!
                      ein Zeichen zur Ausgabe vorbereiten
3082  E6 7F           AND     7FH             ;Bit 7 löschen
3084  E5              PUSH    HL              ;Stringzeiger auf Stack
3085  CD 57 31        CALL    3157H           ;Zeichen ausgeben
3088  E1              POP     HL              ;Stringzeiger laden
3089  23              INC     HL              ;Stringzeiger + 1
308A  C9              RET
```

```
                        Gepufferte Ausgabe von Zeichen
308B  F5          PUSH  AF              ;Zeichen sichern
308C  3A 3B 78    LD    A,(783BH)       ;I/O - Latch-Byte laden
308F  CB 5F       BIT   3,A             ;System im Grafik-Modus ?
3091  28 17       JR    Z,30AAH         ;nein!

3093  E6 F7       AND   0F7H            ;Bit 3 löschen
3095  32 3B 78    LD    (783BH),A       ;I/O - Latch-Byte speichern
3098  32 00 68    LD    (6800H),A       ;umschalten in Text-Modus
309B  01 00 02    LD    BC,512          ;Bildspeicher für Text-Modus
309E  21 00 70    LD    HL,7000H        ;löschen (512 Byte)
30A1  CD BE 3E    CALL  3EBEH           ;Zeichen löschen
30A4  23          INC   HL              ;nächste Adresse
30A5  0B          DEC   BC              ;Zähler - 1
30A6  79          LD    A,C             ;= 0 ? (fertig)
30A7  B0          OR    B
30A8  20 F7       JR    NZ,30A1H        ;nein, nächstes Zeichen

30AA  F1          POP   AF              ;auszugebendes Zeichen laden
30AB  21 39 78    LD    HL,7839H        ;Flag 2 adressieren
30AE  CB 6E       BIT   5,(HL)          ;Initialisierungs-Flag gesetzt ?
30B0  CA 06 31    JP    Z,3106H         ;nein, Direktausgabe d. Zeichens
30B3  FE 20       CP    20H             ;Ist es ein Steuerzeichen ?
30B5  D2 C0 30    JP    NC,30C0H        ;nein!
                        warten, bis Druckpuffer vollständig ausgegeben
30B8  F5          PUSH  AF              ;Zeichen sichern
30B9  3A AF 7A    LD    A,(7AAFH)       ;Pufferzähler laden
30BC  B7          OR    A               ;= 0 ?
30BD  20 FA       JR    NZ,30B9H        ;nein, warten!
30BF  F1          POP   AF              ;Zeichen wieder laden

30C0  F3          DI                    ;Interupts ausschalten
30C1  2A B0 7A    LD    HL,(7AB0H)      ;Pufferzeiger laden
30C4  77          LD    (HL),A          ;Zeichen in Puffer übertragen
30C5  23          INC   HL              ;Pufferzeiger + 1
30C6  22 B0 7A    LD    (7AB0H),HL      ;und zurückspeichern
30C9  21 AF 7A    LD    HL,7AAFH        ;Pufferzähler adressieren
30CC  34          INC   (HL)            ;+ 1
30CD  F5          PUSH  AF              ;Zeichen sichern
30CE  3A A6 78    LD    A,(78A6H)       ;Zeiger auf Bildausgabe-Spalte
30D1  86          ADD   A,(HL)          ;+ Anzahl Zeichen im Puffer
30D2  32 AE 7A    LD    (7AAEH),A       ;= Positions-Zeiger in Zeile
30D5  F1          POP   AF              ;Zeichen wieder laden
30D6  FB          EI                    ;Interupts wieder einschalten
```

```
30D7  FE 20        CP    20H          ;war es ein Steuerzeichen ?
30D9  DA E3 30     JP    C,30AEH      ;ja!
30DC  3E 14        LD    A,20         ;maximal 20 Zeichen im Druckpuffer
30DE  BE           CP    (HL)         ;Grenze überschritten
30DF  DA DE 30     JP    C,30DEH      ;ja, warten bis Puffer leerer
30E2  C9           RET
30E3  AF           XOR   A            ;warten, bis Puffer leer ist
30E4  BE           CP    (HL)
30E5  20 FD        JR    NZ,30E4H
30E7  C9           RET                ;fertig

                   ************************************

                   nach einem Interrupt Druckpuffer ausgeben
30E8  3A AF 7A     LD    A,(7AAFH)    ;Pufferzähler laden
30EB  B7           OR    A            ;Puffer leer?
30EC  C8           RET   Z            ;ja, fertig
30ED  47           LD    B,A          ;in B als Schleifenzähler
30EE  21 B2 7A     LD    HL,7AB2H     ;Puffer-Anfangsadresse laden
30F1  E5           PUSH  HL           ;und auf den Stack
30F2  7E           LD    A,(HL)       ;Zeichen aus Puffer laden
30F3  23           INC   HL           ;Pufferadresse + 1
30F4  E5           PUSH  HL           ;Pufferadresse auf Stack
30F5  C5           PUSH  BC           ;Zeichenzähler auf Stack
30F6  CD 06 31     CALL  3106H        ;Zeichen ausgeben
30F9  C1           POP   BC           ;Zeichenzähler laden
30FA  E1           POP   HL           ;Pufferadresse laden
30FB  10 F5        DJNZ  30F2H        ;Puffer leer ?
30FD  E1           POP   HL           ;ja, Pufferanfangsadresse laden
30FE  22 B0 7A     LD    (7AB0H),HL   ;als neuen Pufferzeiger speichern
3101  AF           XOR   A            ;Pufferzähler = 0
3102  32 AF 7A     LD    (7AAFH),A
3105  C9           RET

                   ************************************

                   Steuerung der Ausgabe eines Zeichens
3106  CD 0D 03     CALL  030DH        ;Zeichen an Cursorpos. wiederherst.
3109  B7           OR    A            ;Auszug. Zeichen = 0 ?
310A  28 04        JR    Z,3110H      ;ja, CR von RDLINE!
310C  FE 0D        CP    0DH          ;ist es ein Carrige-Return ?
310E  20 4A        JR    NZ,315AH     ;nein, Zeichen ausgeben

3110  F5           PUSH  AF           ;Zeichen sichern
```

```
3111  2A 20 78      LD    HL,(7820H)     ;Cursoradresse laden
3114  3A A6 78      LD    A,(78A6H)      ;Spaltenzeiger laden
3117  4F            LD    C,A            ;in BC übertragen
3118  AF            XOR   A
3119  47            LD    B,A
311A  32 A6 78      LD    (78A6H),A      ;Spaltenzeiger auf Spalte 0
311D  ED 42         SBC   HL,BC          ;Cursor-Adresse auf Zeilenanfang
311F  01 20 00      LD    BC,32          ;+ 1 Zeile (32 Zeichen)
3122  09            ADD   HL,BC
3123  7C            LD    A,H            ;Adresse außerhalb des Bildes ?
3124  FE 72         CP    72H
3126  F4 F3 33      CALL  P,33F3H        ;ja, Bild 1 Zeile hochrollen
3129  22 20 78      LD    (7820H),HL     ;neue Cursor-Adresse speichern
312C  CD 53 00      CALL  0053H          ;Zeichen aus Cursor-Position sich.
312F  F1            POP   AF             ;auszugebendes Zeichen laden
3130  B7            OR    A              ;CR von RDLINE (03E3H) ?
3131  C8            RET   Z              ;ja, fertig
3132  CD A8 33      CALL  33A8H          ;Status der Zeile ermitteln
3135  FE 80         CP    80H            ;Einzel-Zeile?
3137  C8            RET   Z              ;ja, fertig
3138  FE 81         CP    81H            ;Erste einer Doppelzeile?
313A  20 05         JR    NZ,3141H       ;nein, Folgezeile
313C  3D            DEC   A              ;beide Zeilen zu Einzelzeilen
313D  77            LD    (HL),A         ;umwandeln
313E  23            INC   HL
313F  77            LD    (HL),A
3140  C9            RET                  ;fertig
3141  3E 80         LD    A,80H          ;Folgezeile in Einzelzeile
3143  77            LD    (HL),A         ;umwandeln
3144  C9            RET                  ;fertig

                    ************************************

                    Zeichen ausgeben, bzw. Steuerfunktionen ausführen
                    Einsprung ist 3157 bzw. 315A
3145  CB 77         BIT   6,A            ;Invertiertes alphan. Zeichen ?
3147  28 04         JR    Z,314DH        ;nein, Blockgrafik!
3149  C3 60 3F      JP    3F60H          ;Invertierte Darstellung abhängig
314C  00            NOP                  ;vom Hintergrund
314D  E6 BF         AND   BFH            ;Bits 4,5,6 löschen
314F  47            LD    B,A            ;Grafikzeichen in B
3150  3A 46 78      LD    A,(7846H)      ;Farbcode laden
3153  B0            OR    B              ;mit Grafikzeichen kombinieren
3154  47            LD    B,A            ;Zeichen in B
```

```
3155  18 5F          JR    31B6H       ;und ausgeben

3157  CD 0D 03       CALL  030DH       ;Zeichen an Cursorpos. wiederherst.
315A  B7             OR    A           ;Bit 7 des auszug. Zeichens = 1 ?
315B  FA 45 31       JP    M,3145H     ;ja, Sprung
315E  FE 0D          CP    0DH         ;Carriage-Return ?
3160  C8             RET   Z           ;ja, fertig!
3161  FE 08          CP    8           ;BACKSPACE ?
3163  CA 27 32       JP    Z,3227H     ;ja!
3166  FE 1B          CP    1BH         ;Cursor hoch ?
3168  CA 53 32       JP    Z,3253H     ;ja!
316B  FE 0A          CP    0AH         ;Cursor nach unten ?
316D  CA 6D 32       JP    Z,326DH     ;ja!
3170  FE 08          CP    8           ;Cursor links ?
3172  CA 27 32       JP    Z,3227H     ;ja!
3175  FE 09          CP    9           ;Cursor rechts ?
3177  CA B8 31       JP    Z,31B8H     ;ja!
317A  FE 01          CP    1           ;BREAK ?
317C  C8             RET   Z           ;ja, fertig !
317D  FE 7F          CP    7FH         ;RUBOUT ?
317F  CA CB 33       JP    Z,33CBH     ;ja!
3182  FE 15          CP    15H         ;INSERT ?
3184  CA C6 32       JP    Z,32C6H     ;ja!
3187  FE 18          CP    18H         ;Pfeil links ?
3189  CA 27 32       JP    Z,3227H     ;ja!
318C  FFE 19         CP    19H         ;Pfeil rechts ?
318E  CA B8 31       JP    Z,31B8H     ;ja!
3191  FE 1B          CP    1BH         ;Pfeil hoch ?
3193  CA 53 32       JP    Z,3253H     ;ja!
3196  FE 1C          CP    1CH         ;Cursor an Bildanfang ?
3198  CA 87 32       JP    Z,3287H     ;ja!
319B  FE 1D          CP    1DH         ;Cursor an Zeilenanfang ?
319D  CA B4 32       JP    Z,32B4H     ;ja!
31A0  FE 1F          CP    1FH         ;Bild löschen ?
31A2  CA 92 32       JP    Z,3292H     ;ja!
31A5  FE 20          CP    20H         ;restliche Steuerzeichen ignorieren
31A7  F8             RET   M
31A8  C3 CA 3E       JP    3ECAH       ;weiter bei 3ECA (Rucksack)
31AB  21 38 78       LD    HL,7838H    ;Flag I adressieren
31AE  CB 4E          BIT   1,(HL)      ;INVERSE-Flag gesetzt ?
31B0  E1             POP   HL          ;Stack bereinigen
31B1  28 02          JR    Z,31B5H     ;nein!
31B3  F6 40          OR    40H         ;ja, Zeichen invertieren
31B5  47             LD    B,A         ;Zeichen in B
```

```
31B6  78
31B7  77
31B8  CD BF 31
31BB  CD 50 00
31BE  C9
```

```
LD      A,B             ;Zeichen in A übertragen
LD      (HL),A          ;auf Bildschirm ausgeben
CALL    31BFH           ;Cursor eine Stelle vorsetzen
CALL    0050H           ;Zeiche an Cursorpos. sichern
RET
```

```
*************************************
```

```
Cursor eine Zeichenposition vorsetzen
31BF  3A A6 78  LD      A,(78A6H)       ;Spaltenzeiger laden
31C2  3C        INC     A               ;+ 1
31C3  FE 20     CP      32              ;am Anfang der nächsten Zeile ?
31C5  20 2B     JR      NZ,31F2H        ;nein!
31C7  CD A8 33  CALL    33A8H           ;Zeilenstatus ermitteln
31CA  FE 81     CP      81H             ;erste von zwei Zeilen ?
31CC  28 23     JR      Z,31F1H         ;ja!
31CE  B7        OR      A               ;zweite von zwei Zeilen ?
31CF  20 35     JR      NZ,3206H        ;nein!
31D1  47        LD      B,A             ;Status in B
31D2  3A 39 78  LD      A,(7839H)       ;Flag 2 laden
31D5  CB 47     BIT     0,A             ;CR-Flag gesetzt ?
31D7  78        LD      A,B             ;Status wieder in A
31D8  C8        RET     Z               ;ja, maximal zwei Zeilen!
31D9  AF        XOR     A               ;nächste Zeile = Folgezeile
31DA  23        INC     HL
31DB  77        LD      (HL),A          ;(= 00)
31DC  23        INC     HL              ;HL auf Status d. n. Zeile
31DD  E5        PUSH    HL              ;und merken
31DE  ED 4B A4 78  LD   BC,(78A4H)      ;Ist dies die letzte Zeile?
31E2  0B        DEC     BC              ;(Diese Routine kollidiert)
31E3  0B        DEC     BC              ;(mit Programmen, die nicht)
31E4  B7        OR      A               ;(standardmäßig beginnen)
31E5  ED 42     SBC     HL,BC
31E7  E1        POP     HL              ;Status der Zeile laden
31E8  30 07     JR      NC,31F1H        ;ja, letzte Zeile
31EA  7E        LD      A,(HL)          ;Status der Zeile = 00 ?
31EB  B7        OR      A               ;wenn ja, Folgezeile
31EC  20 03     JR      NZ,31F1H        ;nein!
31EE  3E 80     LD      A,80H           ;Zeile als Einzelzeile kennz.
31F0  77        LD      (HL),A
31F1  AF        XOR     A               ;Spaltenzeiger = 0
31F2  32 A6 78  LD      (78A6H),A       ;A in Spaltenzeiger übertragen
31F5  2A 20 78  LD      HL,(7820H)      ;Cursor-Adresse laden
```

```
31F8  01 01 00        LD    BC,1          ;+ 1
31FB  09              ADD   HL,BC
31FC  7C              LD    A,H           ;außerhalb des Bildes ?
31FD  FE 72           CP    72H
31FF  F4 F3 33        CALL  P,33F3H       ;ja, eine Zeile hochrollen
3202  22 20 78        LD    (7820H),HL    ;Cursor-Adresse speichern
3205  C9              RET                 ;fertig

3206  F5              PUSH  AF            ;Status merken
3207  ED 5B 20 78     LD    DE,(7820H)    ;Cursor-Adresse laden
320B  13              INC   DE            ;+ 1
320C  7A              LD    A,D           ;außerhalb des Bildes ?
320D  FE 72           CP    72H
320F  28 10           JR    Z,3221H       ;ja!
3211  E5              PUSH  HL            ;Status-Adresse auf Stack
3212  21 39 78        LD    HL,7839H      ;Flag 2 adressieren
3215  CB 46           BIT   0,(HL)        ;CR - Flag gesetzt ?
3217  20 07           JR    NZ,3220H      ;ja!
3219  CB 66           BIT   4,(HL)        ;INPUT - Flag gesetzt ?
321B  20 03           JR    NZ,3220H      ;ja!
321D  CD 2C 33        CALL  332CH         ;eine Zeile zurückrollen
3220  E1              POP   HL            ;Status-Adresse laden
3221  F1              POP   AF            ;Zeilenstatus laden
3222  3C              INC   A             ;Status = 01 setzen
3223  77              LD    (HL),A        ;= zweizeilig!
3224  C3 D9 31        JP    31D9H

                      ************************************

                      Cursor ein Zeichen nach links
3227  3A A6 78        LD    A,(78A6H)     ;Spaltenzeiger laden
322A  3D              DEC   A             ;- 1
322B  F2 35 32        JP    P,3235H       ;noch dieselbe Zeile!
322E  CD A8 33        CALL  33A8H         ;Zeilenstatus ermitteln
3231  B7              OR    A             ;ist dies eine Folgezeile ?
3232  C0              RET   NZ            ;nein, weiter zurück geht nicht
3233  3E 1F           LD    A,31          ;Spaltenzeiger auf letzte Spalte
3235  32 A6 78        LD    (78A6H),A     ;und speichern
3238  01 01 00        LD    BC,1          ;Cursoradresse - 1
323B  2A 20 78        LD    HL,(7820H)
323E  AF              XOR   A
323F  ED 42           SBC   HL,BC
3241  7C              LD    A,H           ;außerhalb des Bildes ?
3242  FE 70           CP    70H
```

```
3244  DA 4E 32        JP      C,324EH         ;ja!
3247  22 20 78        LD      (7820H),HL      ;neue Cursor-Adresse zurück
324A  CD 53 00        CALL    0053H           ;Zeichen an Cursorpos. sichern
324D  C9              RET
324E  AF              XOR     A               ;Spaltenzeiger = 0 (1. Spalte)
324F  32 A6 78        LD      (78A6H),A
3252  C9              RET

                      ************************************

                      Cursor eine Zeile nach oben
3253  21 39 78        LD      HL,7839H        ;Flag 2 adressieren
3256  CB 66           BIT     4,(HL)          ;INPUT-Flag gesetzt ?
3258  C0              RET     NZ              ;ja, unzulässig!
3259  01 20 00        LD      BC,32           ;Länge einer Zeile
325C  2A 20 78        LD      HL,(7820H)      ;Cursor-Adresse laden
325F  AF              XOR     A               ;Carry-Flag löschen
3260  ED 42           SBC     HL,BC           ;Cursor-Adresse - 1 Zeile
3262  7C              LD      A,H             ;außerhalb des Bildschirms ?
3263  FE 70           CP      70H
3265  F8              RET     M               ;ja, geht nicht !
3266  22 20 78        LD      (7820H),HL      ;Cursor-Adresse speichern
3269  CD 53 00        CALL    0053H           ;Zeichen an Cursorpos. sichern
326C  C9              RET

                      ************************************

                      Cursor eine Zeile nach unten
326D  21 39 78        LD      HL,7839H        ;Flag 2 adressieren
3270  CB 66           BIT     4,(HL)          ;INPUT-Flag gesetzt ?
3272  C0              RET     NZ              ;ja, nicht zulässig!
3273  01 20 00        LD      BC,32           ;Länge einer Zeile laden
3276  2A 20 78        LD      HL,(7820H)      ;Cursor-Adresse laden
3279  09              ADD     HL,BC           ;+ eine Zeile
327A  7C              LD      A,H             ;außerhalb des Bildschirms ?
327B  FE 72           CP      72H
327D  F4 24 34        CALL    P,3424H         ;ja, eine Zeile hochrollen
3280  22 20 78        LD      (7820H),HL      ;Cursor-Adresse speichern
3283  CD 53 00        CALL    0053H           ;Zeichen an Cursorpos. sichern
3286  C9              RET

                      ************************************

                      Cursor an den Bildschirmanfang
```

```
3287  21 00 70        LD      HL,7000H        ;Bildanfangsadresse laden
328A  22 20 78        LD      (7820H),HL      ;in Cursor-Adresse
328D  AF              XOR     A               ;Spaltenzeiger = 0
328E  32 A6 78        LD      (78A6H),A
3291  C9              RET

                      ************************************

                      Bildschirm löschen
3292  21 00 70        LD      HL,7000H        ;Bildanfangsadresse laden
3295  22 20 78        LD      (7820H),HL      ;in Cursor-Adresse
3298  01 00 02        LD      BC,512          ;Länge des Textspeichers
329B  CD BE 3E        CALL    3EBEH           ;ein Zeichen löschen
329E  23              INC     HL              ;Bildadresse + 1
329F  0B              DEC     BC              ;Länge - 1
32A0  79              LD      A,C             ;am Bildende ?
32A1  B0              OR      B
32A2  20 F7           JR      NZ,329BH        ;nein, nächstes Zeichen
32A4  AF              XOR     A               ;Spaltenzeiger = 0
32A5  32 A6 78        LD      (78A6H),A
32A8  06 10           LD      B,16            ;Zähler = 16 (Anzahl Zeilen)
32AA  3E 80           LD      A,80H           ;Einzeilen-status laden
32AC  21 D7 7A        LD      HL,7AD7H        ;Statusbyte 1. Zeile adressieren
32AF  77              LD      (HL),A          ;Status = einzeilig
32B0  23              INC     HL              ;Statusbyte nächste Zeile
32B1  10 FC           DJNZ    32AFH           ;alle Zeilen bearbeitet ?
32B3  C9              RET                     ;ja, fertig!

                      ************************************

                      Cursor an Zeilenanfang
32B4  2A 20 78        LD      HL,(7820H)      ;Cursor-Adresse laden
32B7  3A A6 78        LD      A,(78A6H)       ;Spaltenzeiger laden
32BA  4F              LD      C,A             ;in BC übertragen
32BB  AF              XOR     A
32BC  47              LD      B,A
32BD  32 A6 78        LD      (78A6H),A       ;Spaltenzeiger = 0
32C0  ED 42           SBC     HL,BC           ;Cursor-Adresse - Spaltenzeiger
32C2  22 20 78        LD      (7820H),HL      ;neue Cursor-Adresse sichern
32C5  C9              RET

                      ************************************

                      INSERT - Funktion
```

```
32C6  CD A8 33     CALL   33A8H          ;Status der Zeile ermitteln
32C9  FE 81        CP     81H            ;erste von zwei Zeilen ?
32CB  28 31        JR     Z,32FEH        ;ja!
32CD  3A A6 78     LD     A,(78A6H)      ;Spaltenzeiger laden
32D0  FE 1F        CP     31             ;am Ende der Zeile ?
32D2  28 25        JR     Z,32F9H        ;ja!
32D4  4F           LD     C,A            ;Spaltenzeiger in BC
32D5  AF           XOR    A
32D6  47           LD     B,A
32D7  2A 20 78     LD     HL,(7820H)     ;Cursoradresse laden
32DA  ED 42        SBC    HL,BC          ;- Spaltenzeiger = Zeilenanfang
32DC  01 1F 00     LD     BC,31          ;letztes Zeichen der Zeile adress.
32DF  09           ADD    HL,BC
32E0  CD E9 3E     CALL   3EE9H          ;letztes Zeichen der Zeile testen
32E3  20 14        JR     NZ,32F9H       ;ungleich Leerzeichen
32E5  E5           PUSH   HL             ;Adresse letztes Zeichen
32E6  D1           POP    DE             ;in DE
32E7  2B           DEC    HL             ;HL = vorletztes Zeichen
32E8  3A A6 78     LD     A,(78A6H)      ;Spaltenzeiger laden
32EB  4F           LD     C,A            ;Zeilenlänge - 1 - Spalte
32EC  3E 1F        LD     A,31           ;in BC für Block-Move LDDR
32EE  91           SUB    C
32EF  4F           LD     C,A
32F0  ED B8        LDDR                  ;ab Cursorpos.1 Zeichen rechts sch.
32F2  CD F6 3E     CALL   3EF6H          ;Leerzeichen einfügen
32F5  32 3C 78     LD     (783CH),A      ;Zeichen sichern
32F8  C9           RET

32F9  CD A8 33     CALL   33A8H          ;Status der Zeile ermitteln
32FC  B7           OR     A              ;Folgezeile ?
32FD  C8           RET    Z              ;ja, fertig
32FE  FE 80        CP     80H            ;Einzel-Zeile ?
3300  28 1E        JR     Z,3320H        ;ja!
3302  3A A6 78     LD     A,(78A6H)      ;Spaltenzeiger laden
3305  4F           LD     C,A            ;in BC übertragen
3306  AF           XOR    A
3307  47           LD     B,A
3308  2A 20 78     LD     HL,(7820H)     ;Cursor-Adresse laden
330B  ED 42        SBC    HL,BC          ;- Spaltenzeiger = Zeilenanfang
330D  01 3F 00     LD     BC,63          ;+ 63
3310  09           ADD    HL,BC          ;= Ende der Doppelzeile
3311  CD E9 3E     CALL   3EE9H          ;letztes Zeichen = leer ?
3314  C0           RET    NZ             ;nein, kein Einfügen möglich
3315  E5           PUSH   HL             ;Endadresse
```

```
3316  D1          POP   DE          ;in DE
3317  2B          DEC   HL          ;HL = Endadresse - 1
3318  3A A6 78    LD    A,(78A6H)   ;Anzahl zu verschiebender Zeichen
331B  4F          LD    C,A         ;ermitteln und in BC
331C  3E 3F       LD    A,63        ;(= 63 - Spaltenzeiger)
331E  18 CE       JR    32EEH       ;Doppelzeile 1 Zeichen rechts sch.

3320  E5          PUSH  HL          ;Statusadresse auf den Stack
3321  CD 2C 33    CALL  332CH       ;Bildschirm 1 Zeile rollen
3324  E1          POP   HL          ;Statusadresse laden
3325  3E 81       LD    A,81H       ;Status der 1. Zeile = 81 setzen
3327  77          LD    (HL),A      ;= zweizeilig
3328  23          INC   HL          ;Status der neuen Zeile
3329  AF          XOR   A           ;als Folgezeile deklarieren
332A  77          LD    (HL),A      ;(= 00)
332B  C9          RET

                  ************************************

                  Bildschirm ab Cursor-Position eine Zeile
                  nach unten rollen
332C  2A 20 78    LD    HL,(7820H)  ;Cursor-Adresse laden
332F  7C          LD    A,H         ;Ist es die letzte Zeile ?
3330  FE 71       CP    71H
3332  20 2B       JR    NZ,335FH    ;nein!
3334  7D          LD    A,L         ;in der unteren Bildhälfte
3335  FE E0       CP    0E0H        ;auch das 2. Adressbyte prüfen
3337  DA 5F 33    JP    C,335FH     ;nicht letzte Zeile!
333A  3A A6 78    LD    A,(78A6H)   ;Spaltenzeiger laden
333D  F5          PUSH  AF          ;und auf dem Stack merken
333E  3A D7 7A    LD    A,(7AD7H)   ;Status der 1. Zeile laden
3341  FE 81       CP    81H         ;ist es eine Doppelzeile ?
3343  20 08       JR    NZ,334DH    ;nein, nur 1 Zeile rollen
3345  E5          PUSH  HL          ;Cursor-Adresse auf den Stack
3346  CD F3 33    CALL  33F3H       ;Bild 1 Zeile hochrollen
3349  E1          POP   HL          ;Cursor-Adresse laden
334A  CD 17 03    CALL  0317H       ;Cursor-Adresse - 1 Zeile
334D  E5          PUSH  HL          ;Cursor-Adresse auf den Stack
334E  CD F3 33    CALL  33F3H       ;Bild 1 Zeile hochrollen
3351  E1          POP   HL          ;Cursor-Adresse laden
3352  CD 17 03    CALL  0317H       ;Cursor-Adresse - 1 Zeile
3355  F1          POP   AF          ;Spaltenzeiger laden
3356  32 A6 78    LD    (78A6H),A   ;und alten Wert zurückschreiben
3359  D1          POP   DE          ;Rücksprungadr. vom Stack
```

```
335A  E1           POP   HL          ;Statusadresse vom Stack
335B  2B           DEC   HL          ;Statusadresse - 1 Zeile
335C  E5           PUSH  HL          ;Statusadresse auf den Stack
335D  D5           PUSH  DE          ;Rücksprungadr. auf den Stack
335E  C9           RET               ;fertig

335F  3A A6 78     LD    A,(78A6H)   ;Spaltenzeiger laden
3362  4F           LD    C,A         ;in BC übertragen
3363  AF           XOR   A
3364  47           LD    B,A
3365  ED 42        SBC   HL,BC       ;Cursoradr. - Spaltenzeiger
3367  01 40 00     LD    BC,64       ;+ 64
336A  09           ADD   HL,BC       ;= Anfangsadresse der übern. Zeile
336B  E5           PUSH  HL          ;merken
336C  EB           EX    DE,HL       ;und in DE
336D  21 00 72     LD    HL,7200H    ;Bildendadresse + 1
3370  ED 52        SBC   HL,DE       ;- DE = Anzahl zu versch. Bytes
3372  E5           PUSH  HL          ;merken
3373  C1           POP   BC          ;Bytezähler in BC laden
3374  21 DF 71     LD    HL,71DFH    ;Endadresse vorletzte Zeile
3377  11 FF 71     LD    DE,71FFH    ;Endadresse letzte Zeile
337A  79           LD    A,C         ;Bytezähler = 0 ?
337B  B0           OR    B
337C  28 02        JR    Z,3380H     ;ja, keine Verschiebung
337E  ED B8        LDDR              ;nein, Bild eine Zeile nach unten
3380  E1           POP   HL          ;Zeilenadresse der neuen Zeile lad.
3381  CD 02 3F     CALL  3F02H       ;Löschzeichen ermitteln
3384  00           NOP
3385  12           LD    (DE),A      ;neue Zeile löschen
3386  1B           DEC   DE
3387  10 FC        DJNZ  3385H
3389  CD A8 33     CALL  33A8H       ;Status der Zeile ermitteln
338C  E5           PUSH  HL          ;Statusadresse in BC
338D  C1           POP   BC
338E  21 E6 7A     LD    HL,7AE6H    ;Statusadresse letzte Zeile
3391  E5           PUSH  HL          ;auf den Stack
3392  B7           OR    A
3393  ED 42        SBC   HL,BC       ;- Statusadresse akt. Zeile
3395  E5           PUSH  HL          ;Differenz = zu verschiebende Bytes
3396  C1           POP   BC          ;in BC als Bytezähler
3397  E1           POP   HL          ;HL = Statusadresse vorl. Zeile
3398  E5           PUSH  HL          ;DE = Statusadresse letzte Zeile
3399  D1           POP   DE
339A  2B           DEC   HL
```

```
339B  ED B8        LDDR                ;Statusbyte eine Zeile nach unten
339D  3A E6 7A     LD    A,(7AE6H)     ;wenn letzte Zeile keine
33A0  FE 81        CP    81H           ;Doppelzeile
33A2  C0           RET   NZ            ;dann fertig und zurück
33A3  2A 20 78     LD    HL,(7820H)    ;sonst Cursor-Adresse laden
33A6  18 B7        JR    335FH         ;und noch eine Zeile rollen

                   ************************************

                   Zeilenstatus ermitteln
                   Ausg.: A = Zeilenstatus
                          HL = Statusadresse
33A8  3A A6 78     LD    A,(78A6H)     ;Spaltenzeiger laden
33AB  4F           LD    C,A           ;in BC übertragen
33AC  AF           XOR   A
33AD  47           LD    B,A
33AE  2A 20 78     LD    HL,(7820H)    ;Cursoradresse laden
33B1  ED 42        SBC   HL,BC         ;- Spaltenzeiger = Zeilenanfang
33B3  E5           PUSH  HL            ;Zeilenadresse in BC
33B4  C1           POP   BC
33B5  78           LD    A,B           ;Zeilennummer ermitteln
33B6  E6 0F        AND   0FH           ;= Zeilenadresse - 7000H
33B8  CB 3F        SRL   A             ;/ 2
33BA  47           LD    B,A
33BB  CB 19        RR    C             ;= ZNr. im linken Halbbyte von C
33BD  CB 39        SRL   C             ;ins rechte Halbbyte schieben
33BF  CB 39        SRL   C
33C1  CB 39        SRL   C
33C3  CB 39        SRL   C
33C5  21 D7 7A     LD    HL,7AD7H      ;Anfangsadresse der Statustabelle
33C8  09           ADD   HL,BC         ;+ Zeilennummer
33C9  7E           LD    A,(HL)        ;Zeilenstatus laden
33CA  C9           RET

                   ************************************

                   RUBOUT - Funktion
33CB  CD A8 33     CALL  33A8H         ;Zeilenstatus ermitteln
33CE  FE 81        CP    81H           ;erste einer Doppelzeile ?
33D0  2A 20 78     LD    HL,(7820H)    ;Cursor-Adresse laden
33D3  E5           PUSH  HL            ;in DE
33D4  D1           POP   DE
33D5  23           INC   HL            ;HL auf nächste Zeichenposition
33D6  3A A6 78     LD    A,(78A6H)     ;Spaltenzeiger laden
```

```
33D9  4F          LD    C,A         ;in C übertragen
33DA  29 13       JR    Z,33EFH     ;bei Doppelzeile, Sprung
33DC  FE 1F       CP    31          ;Ende der Zeile ?
33DE  28 08       JR    Z,33E8H     ;ja, nur dieses Zeichen löschen
33E0  3E 1F       LD    A,31        ;Länge einer Zeile
33E2  91          SUB   C           ;- Spaltenzeiger
33E3  4F          LD    C,A         ;in BC als Zähler
33E4  AF          XOR   A
33E5  47          LD    B,A
33E6  ED B0       LDIR              ;Zeile ein Zeichen verkürzen
33E8  CD F6 3E    CALL  3EF6H       ;Leerzeichen ans Zeilenende
33EB  CD 50 00    CALL  0050H       ;Zeichen an Cursorposition sichern
33EE  C9          RET               ;fertig

33EF  3E 3F       LD    A,63        ;Länge von zwei Zeilen laden
33F1  18 EF       JR    33E2        ;über zwei Zeilen verkürzen

                  ************************************

                  Bildschirm eine Zeile nach oben rollen.
                  Die letzte Zeile wird mit Leerzeichen gefüllt.
33F3  11 00 70    LD    DE,7000H    ;Adresse der 1. Zeile in DE
33F6  21 20 70    LD    HL,7020H    ;Adresse der 2. Zeile in HL
33F9  01 E0 01    LD    BC,480      ;Bytezähler = 15 Zeilen
33FC  ED B0       LDIR              ;Bild eine Zeile nach oben rollen
33FE  CD 02 3F    CALL  3F02H       ;vorbereiten des Löschens
3401  00          NOP               ;A = Leerz., B = Bytes/Zeile
3402  12          LD    (DE),A      ;letzte Zeile löschen
3403  13          INC   DE
3404  10 FC       DJNZ  3402H
3406  21 D7 7A    LD    HL,7AD7H    ;Statustabelle ebenfalls
3409  E5          PUSH  HL          ;eine Zeile hochrollen
340A  D1          POP   DE          ;DE = Status Zeile 1
340B  23          INC   HL          ;HL = Status Zeile 2
340C  01 0F 00    LD    BC,15       ;BC = Zeilenzähler
340F  ED B0       LDIR              ;von DE nach HL (Länge 15)
3411  1A          LD    A,(DE)      ;Status der letzten Zeile laden
3412  FE 81       CP    81H         ;war dies eine Doppelzeile ?
3414  20 03       JR    NZ,3419H    ;nein, letzte Zeile = Einzelzeie
3416  AF          XOR   A           ;ja, letzte Zeile = Folgezeile
3417  18 02       JR    341BH
3419  3E 80       LD    A,80H       ;X'80' = Kennung f. Einzelzeile
341B  12          LD    (DE),A      ;neue Kennung für letzte Zeile
341C  AF          XOR   A           ;Spaltenzeiger = 0
```

```
341D  32 A6 78        LD    (78A6H),A
3420  21 E0 71        LD    HL,71E0H      ;HL = Anfang der letzten Zeile
3423  C9              RET                 ;fertig

                      Abhängig vom Status der ersten Zeile
                      entweder eine oder zwei Zeilen hochrollen.
3424  3A D7 7A        LD    A,(7AD7H)     ;Status 1. Zeile laden
3427  FE 81           CP    81H           ;= Doppelzeile ?
3429  CC F3 33        CALL  Z,33F3H       ;ja, eine Zeilen rollen
342C  CD F3 33        CALL  33F3H         ;eine Zeile rollen
342F  C9              RET

                      ************************************

                      Bei Zeicheneingabe akustisches Signal ausgeben
3430  21 39 78        LD    HL,7839H      ;Flag 2 adressieren
3433  B7              OR    A             ;Zeichen eingegeben?
3434  20 0B           JR    NZ,3441H      ;ja!
3436  CB CE           SET   1,(HL)        ;nein, BUZZER-Flag setzen
3438  01 FF 03        LD    BC,03FFH      ;Warteschleife
343B  0B              DEC   BC            ;bis zum Ende des vert.Sync-Pulses
343C  79              LD    A,C
343D  B0              OR    B
343E  20 FB           JR    NZ,343BH
3440  C9              RET                 ;fertig

3441  CB 46           BIT   0,(HL)        ;Carriage-Return Flag gesetzt?
3443  C0              RET   NZ            ;ja, fertig
3444  FE 0D           CP    0DH           ;Zeichen = Carriage-Return?
3446  28 06           JR    Z,344EH       ;ja!
3448  FE 01           CP    01H           ;Zeichen = BREAK ?
344A  20 04           JR    NZ,3450H      ;nein!
344C  CB D6           SET   2,(HL)        ;BREAK-Flag setzen
344E  CB C6           SET   0,(HL)        ;Carriage-Return Flag setzen

3450  E5              PUSH  HL            ;Flag 2 - Adresse auf Stack
3451  21 A0 00        LD    HL,00A0H      ;Frequenz laden
3454  01 06 00        LD    BC,6          ;Tondauer laden
3457  CD 5C 34        CALL  345CH         ;Piep-Ton ausgeben
345A  E1              POP   HL            ;Flag 2 - Adresse laden
345B  C9              RET                 ;fertig

                      ************************************
```

```
                        Tonausgabe
                        Eing.: HL = Frequenz
                               BC = Tondauer
345C  3A 3B 78          LD     A,(783BH)      ;I/O Latch-Byte laden
345F  57                LD     D,A            ;in D übertragen
3460  CD 69 34          CALL   3469H          ;Ton ausgeben
3463  0B                DEC    BC             ;Tonlänge - 1
3464  79                LD     A,C            ;= 0 ?
3465  B0                OR     B
3466  20 F8             JR     NZ,3460H       ;nein, weiter ausgeben
3468  C9                RET                   ;ja, fertig

3469  C5                PUSH   BC             ;Tondauer sichern
346A  7A                LD     A,D            ;I/O Latch-Byte in A
346B  EE 21             XOR    21H            ;Bits 0 u. 5 komplementieren
346D  32 00 68          LD     (6800H),A      ;I/O Latch-Byte ausgeben
3470  E5                PUSH   HL             ;Frequenzzähler in BC
3471  C1                POP    BC
3472  0B                DEC    BC             ;untere Halbwelle des Tons bilden
3473  79                LD     A,C
3474  B0                OR     B
3475  20 FB             JR     NZ,3472H
3477  7A                LD     A,D            ;I/O Latch-Byte in A
3478  32 00 68          LD     (6800H),A      ;und ausgeben
347B  E5                PUSH   HL             ;Frequenzzähler in BC
347C  C1                POP    BC
347D  0B                DEC    BC             ;obere Halbwelle des Tons bilden
347E  79                LD     A,C
347F  B0                OR     B
3480  20 FB             JR     NZ,347DH
3482  C1                POP    BC             ;Tondauer laden
3483  C9                RET

                        ************************************

                        Teil der Initialisierungs-Routine
3484  CD A0 3F          CALL   3FA0H          ;Prüfen ob CTRL-Taste gedrückt
                                              ;und Hintergrundfarbe entspr. einst
3487  3E 20             LD     A,' '          ;Grundeinstellung in I/O-Latch
3489  32 3B 78          LD     (783BH),A      ;merken!
348C  32 00 68          LD     (6800H),A      ;und ausgeben
348F  3E 3C             LD     A,60           ;Zeitzähler = 60
3491  32 3A 78          LD     (783AH),A
3494  3E 10             LD     A,16           ;Blinkzähler initialisieren
```

```
3496  32 41 78        LD    (7841H),A
3499  AF              XOR   A              ;Pufferzähler = 0
349A  32 AF 7A        LD    (7AAFH),A
349D  21 B2 7A        LD    HL,7AB2H       ;Pufferzeiger auf Pufferanfang
34A0  22 B0 7A        LD    (7AB0H),HL
34A3  3E C9           LD    A,0C9H         ;RET f. Interrupt Vektor
34A5  C3 37 3E        JP    3E37H          ;Farbe = gelb setzen
34A8  C9              RET                  ;nicht benutzt

                      ************************************

                      CSAVE - Anweisung
                      Ausgabe auf Kassette
34A9  F3              DI                   ;Interrupts ausschalten
34AA  0E F0           LD    C,0F0H         ;Kenner für BASIC-Programm setzen
34AC  CD 58 35        CALL  3558H          ;Vorspann und Programmname ausgeben
34AF  DA FE 3A        JP    C,3AFEH        ;wenn Carry=1, BREAK betätigt.
34B2  E5              PUSH  HL             ;Programmzeiger sichern
34B3  01 9A 01        LD    BC,410         ;3 ms Lücke auf Band
34B6  0B              DEC   BC
34B7  79              LD    A,C
34B8  B0              OR    B
34B9  20 FB           JR    NZ,34B6H
34BB  CD F8 3A        CALL  3AF8H          ;prüfen, ob BREAK-Taste betätigt
34BE  DD 21 23 78     LD    IX,7823H       ;Prüfsummen-Bytes adressieren
34C2  2A A4 78        LD    HL,(78A4H)     ;Programm-Startadresse laden
34C5  7D              LD    A,L            ;LSB Startadresse in A laden,
34C6  CD 11 35        CALL  3511H          ;auf Kassette ausgeben
34C9  DD 77 00        LD    (IX),A         ;und in Prüfsummen-Byte (LSB)
34CC  AF              XOR   A              ;0 in LSB Prüfsummen-Byte
34CD  DD 77 01        LD    (IX+1),A
34D0  7C              LD    A,H            ;MSB Startadresse in A laden
34D1  CD 11 35        CALL  3511H          ;auf Kassette ausgeben
34D4  CD 8E 38        CALL  388EH          ;auf Prüfsumme addieren
34D7  EB              EX    DE,HL          ;Startadresse in DE
34D8  2A F9 78        LD    HL,(78F9H)     ;Programm-Endadresse laden
34DB  7D              LD    A,L            ;LSB Endadresse in A laden
34DC  CD 11 35        CALL  3511H          ;auf Kassette ausgeben
34DF  CD 8E 38        CALL  388EH          ;auf Prüfsumme addieren
34E2  7C              LD    A,H            ;MSB Endadresse laden
34E3  CD 11 35        CALL  3511H          ;auf Kassette ausgeben
34E6  CD 8E 38        CALL  388EH          ;auf Prüfsumme addieren
34E9  CD F8 3A        CALL  3AF8H          ;prüfen, ob BREAK-Taste gedrückt
34EC  1A              LD    A,(DE)         ;Programm-Byte laden
```

```
34ED  13          INC   DE          ;Programmadresse + 1
34EE  CD 11 35    CALL  3511H       ;Programm-Byte auf Kassette
34F1  CD 8E 38    CALL  388EH       ;auf Prüfsumme addieren
34F4  CD F8 3A    CALL  3AF8H       ;prüfen, ob BREAK-Taste gedrückt
34F7  DF          RST   18H         ;Programmende erreicht?
34F8  20 F2       JR    NZ,34ECH    ;nein, nächstes Byte ausgeben
34FA  DD 7E 00    LD    A,(IX)      ;LSB Prüfsumme laden
34FD  CD 11 35    CALL  3511H       ;auf Kassette ausgeben
3500  DD 7E 01    LD    A,(IX+1)    ;MSB Prüfsumme laden
3503  CD 11 35    CALL  3511H       ;auf Kassette ausgeben
3506  06 14       LD    B,20        ;als Endekennung
3508  AF          XOR   A           ;20 Bytes X'00'
3509  CD 11 35    CALL  3511H       ;auf Kassette ausgeben
350C  10 FB       DJNZ  3509H
350E  E1          POP   HL          ;Programmzeiger wieder laden
350F  FB          EI                ;Interrupts wieder einschalten
3510  C9          RET               ;fertig!

                  ************************************

                  Ein Byte auf Kassette schreiben
                  Eing.: A = auszugebendes Byte
3511  F5          PUSH  AF          ;Register-Inhalte sichern
3512  C5          PUSH  BC
3513  E5          PUSH  HL
3514  2E 08       LD    L,8         ;Bitzähler = 8
3516  67          LD    H,A         ;H = auszugebendes Byte
3517  CD 42 35    CALL  3542H       ;Clock-Puls ausgeben
351A  CB 04       RLC   H           ;höchstwertiges Bit in Carry
351C  30 0D       JR    NC,352BH    ;= 0 ? ja, 0-Bit ausgeben
351E  CD 42 35    CALL  3542H       ;nein, 1-Bit ausgeben
3521  CD 42 35    CALL  3542H       ;durch 2 aufeinanderfolg. Clock-P.
3524  2D          DEC   L           ;Bitzähler - 1
3525  20 F0       JR    NZ,3517H    ;> 0? ja, nächstes Bit
3527  E1          POP   HL          ;Registerinhalte wiederherstellen
3528  C1          POP   BC
3529  F1          POP   AF
352A  C9          RET               ;fertig

                  0 - Bit ausgeben
352B  3A 3B 78    LD    A,(783BH)   ;I/O Latch-Byte laden
352E  F6 06       OR    6           ;Bits 1 u. 2 setzen
3530  32 00 68    LD    (6800H),A   ;und ausgeben
3533  06 99       LD    B,153       ;555 us Pause
```

```
3535  10 FE          DJNZ   3535H
3537  E6 F9          AND    0F9H            ;Bits 1 u. 2 wieder löschen
3539  32 00 68       LD     (6800H),A       ;und ausgeben
353C  06 99          LD     B,153           ;555 us Pause
353E  10 FE          DJNZ   353EH
3540  18 E2          JR     3524H           ;fertig

                     Clock-Puls ausgeben
3542  3A 3B 78       LD     A,(783BH)       ;I/O Latch-Byte laden
3545  F6 06          OR     6               ;Bits 1 u. 2 setzen
3547  32 00 68       LD     (6800H),A       ;und ausgeben
354A  06 4C          LD     B,76            ;277 us Pause
354C  10 FE          DJNZ   354CH
354E  E6 F9          AND    0F9H            ;Bits 1 u. 2 wieder löschen
3550  32 00 68       LD     (6800H),A       ;und ausgeben
3553  06 4C          LD     B,776           ;277 us Pause
3555  10 FE          DJNZ   3555H
3557  C9             RET                    ;fertig

                     ************************************

                     Vorspann auf Kassette schreiben
                     (Synchronisation, Vorspann, Kenner, Programmname)
3558  CD 8C 35       CALL   358CH           ;Programmname in Puffer
355B  06 FF          LD     B,255           ;Synchronisations-Bytes ausgeben
355D  3E 80          LD     A,80H           ;= 255 = X'80'
355F  CD 11 35       CALL   3511H           ;Byte ausgeben
3562  CD E8 3A       CALL   3AE8H           ;BREAK-Taste gedrückt ?
3565  D8             RET    C               ;ja, abbrechen!
3566  10 F5          DJNZ   355DH           ;255 Byte Zähler
3568  06 05          LD     B,5             ;5 = X'FE' als Vorspann
356A  3E FE          LD     A,0FEH          ;ausgeben
356C  CD 11 35       CALL   3511H           ;Byte ausgeben
356F  CD E8 3A       CALL   3AE8H           ;BREAK-Taste gedrückt ?
3572  D8             RET    C               ;ja, abbrechen!
3573  10 F5          DJNZ   356AH           ;Byte-Zähler
3575  79             LD     A,C             ;Programm-(Datei) Kenner laden
3576  CD 11 35       CALL   3511H           ;und auf Kassette schreiben
3579  CD E8 3A       CALL   3AE8H           ;BREAK-Taste gedrückt ?
357C  D8             RET    C               ;ja, abbrechen
357D  3A D6 7A       LD     A,(7AD6H)       ;Länge des Namens laden
3580  47             LD     B,A             ;in B als Zähler
3581  11 9D 7A       LD     DE,7A9DH        ;Anfangsadresse des Namens in DE
3584  1A             LD     A,(DE)          ;Zeichen des Namens laden
```

```
3585  13          INC   DE            ;Adresse + 1
3586  CD 11 35    CALL  3511H         ;Zeichen auf Kassette ausgeben
3589  10 F9       DJNZ  3584H         ;Namen vollständig ausgegeben ?
358B  C9          RET                 ;ja, fertig!

                  ************************************

                  Programm- (Datei-) Namen prüfen und in Puffer übernehmen
                  Aufruf von CSAVE, CLOAD, PRINT#, INPUT#
358C  06 10       LD    B,16          ;max. 16 Zeichen
358E  11 9D 7A    LD    DE,7A9DH      ;Anfangsadresse des Puffers laden
3591  7E          LD    A,(HL)        ;Zeichen aus Programmtext laden
3592  FE 3A       CP    ':'           ;Ende des Befehls ?
3594  28 12       JR    Z,35A8H       ;ja!
3596  B7          OR    A             ;Zeilenende ?
3597  28 0F       JR    Z,35A8H       ;ja!
3599  CF          RST   8             ;folgt ein '"' ?
359A  22          DEFB  '"'
359B  7E          LD    A,(HL)        ;Zeichen des Namens in A
359C  B7          OR    A             ;Zeilenende ?
359D  28 09       JR    Z,35A8H       ;ja!
359F  23          INC   HL            ;Programmzeiger + 1
35A0  FE 22       CP    '"'           ;Ende des Strings ?
35A2  28 04       JR    Z,35A8H       ;ja!
35A4  12          LD    (DE),A        ;Zeichen in Puffer übernehmen
35A5  13          INC   DE            ;Pufferadresse + 1
35A6  10 F3       DJNZ  359BH         ;nächstes Zeichen
35A8  AF          XOR   A             ;X'00' als Abschluß
35A9  12          LD    (DE),A        ;in Puffer
35AA  3E 11       LD    A,17          ;Länge des Namens ermitteln
35AC  90          SUB   B
35AD  32 D6 7A    LD    (7AD6H),A     ;und merken
35B0  C9          RET                 ;fertig

                  ************************************

                  Prüfen, ob Lademeldungen ausgegeben werden sollen.
                  Wenn ja, vorbereiten der Ausgabe
35B1  3A 4C 78    LD    A,(784CH)     ;Ausgabe-Flag laden
35B4  B7          OR    A             ;= 0 ?
35B5  C0          RET   NZ            ;nein, Meldungen unterdrücken
35B6  3A 3B 78    LD    A,(783BH)     ;I/O Latch-Byte laden
35B9  CB 5F       BIT   3,A           ;Grafik-Modus ?
35BB  28 0B       JR    Z,35C8H       ;nein!
```

```
35BD  E6 F7       AND   0F7H          ;ja, in Text-Modus umschalten
35BF  32 3B 78    LD    (783BH),A
35C2  32 00 68    LD    (6800H),A
35C5  CD 92 32    CALL  3292H         ;Bildschirm löschen
35C8  21 FF 71    LD    HL,71FFH      ;Cursor auf letzte Bildposition
35CB  22 20 78    LD    (7820H),HL
35CE  3E 1F       LD    A,31          ;Spaltenzeiger auf letzte Spalte
35D0  32 A6 78    LD    (78A6H),A
35D3  3A E5 7A    LD    A,(7AE5H)     ;Status der vorl. Zeile laden
35D6  FE 81       CP    81H           ;Doppelzeile ?
35D8  C0          RET   NZ            ;nein, fertig
35D9  3D          DEC   A             ;letzte und vorl. Zeile als
35DA  32 E5 7A    LD    (7AE5H),A     ;Einzelzeilen kennzeichnen
35DD  32 E6 7A    LD    (7AE6H),A
35E0  C9          RET                 ;fertig

                  ************************************

                  Programmanfang auf der Kassette suchen
35E1  21 42 38    LD    HL,3842H      ;Adresse 'WAITING' - Text laden
35E4  CD F4 37    CALL  37F4H         ;Text ausgeben
                  Einsprung von CLOAD
35E7  CD F8 3A    CALL  3AF8H         ;BREAK-Taste gedrückt ?
35EA  3A 00 68    LD    A,(6800H)     ;I/O-Byte lesen
35ED  CB 77       BIT   6,A           ;Puls von Kassette auswerten
35EF  20 F6       JR    NZ,35E7H      ;kein Puls, zurück!
35F1  CD 8F 37    CALL  378FH         ;Bit lesen
35F4  38 F1       JR    C,35E7H       ;war nichts, zurück
35F6  CB 47       Bit   0,A           ;war das eine '1' ?
35F8  28 F7       JR    Z,35F1H       ;nein, nächstes Bit
35FA  06 07       LD    B,7           ;7 weitere Bits lesen
35FC  CD 8F 37    CALL  378FH         ;Bit lesen
35FF  38 E6       JR    C,35E7H       ;Zeit abgelaufen, noch einmal!
3601  10 F9       DJNZ  35FCH         ;nächstes Bit
3603  FE 80       CP    80H           ;ist es ein SYNC - Byte ?
3605  20 E0       JR    NZ,35E7H      ;nein, weiter suchen
                  Kassette bis hinter SYNC-Bytes vorsetzen
3607  CD 75 37    CALL  3775H         ;Byte lesen
360A  DA E7 35    JP    C,35E7H       ;Zeit abgelaufen, noch einmal!
360D  FE 80       CP    80H           ;SYNC-Byte ?
360F  28 F6       JR    Z,3607H       ;ja, nächstes Byte
                  die nächsten 5 Bytes müssen X'FE' sein
3611  06 04       LD    B,4           ;Zähler = 4, da 1. Byte ber. gelad.
3613  FE FE       CP    0FEH          ;= Vorspann-Byte ?
```

```
3615  C2 E7 35      JP      NZ,35E7H        ;nein, weiter suchen
3618  CD 75 37      CALL    3775H           ;nächstes Byte lesen
361B  DA E7 35      JP      C,35E7H         ;Zeit abgelaufen, zurück!
361E  10 F3         DJNZ    3613H
                    Programmkennung lesen
3620  CD 75 37      CALL    3775H           ;Kennungsbyte lesen
3623  32 D2 7A      LD      (7AD2H),A       ;und speichern
                    Namen lesen und in Puffer übertragen
3626  21 B2 7A      LD      HL,7AB2H        ;Pufferadresse laden
3629  06 12         LD      B,18            ;max. Länge
362B  CD 75 37      CALL    3775H           ;Byte lesen
362E  77            LD      (HL),A          ;und in Puffer übertragen
362F  B7            OR      A               ;= 0 ?
3630  28 06         JR      Z,3638H         ;ja, fertig
3632  23            INC     HL              ;Pufferadresse + 1
3633  10 F6         DJNZ    362BH           ;nächstes Byte des Namens
3635  C3 E7 35      JP      35E7H           ;mehr als 18 Zeichen!
                    FOUND-Meldung ausgeben
3638  21 5A 38      LD      HL,385AH        ;Meldungstext adressieren
363B  CD F4 37      CALL    37F4H           ;Text "FOUND" ausgeben
363E  21 B2 7A      LD      HL,7AB2H        ;Namen im Pufer adressieren
3641  CD 14 38      CALL    3814H           ;und ausgeben.
                    Prüfen, ob gesuchtes Programm
3644  21 B2 7A      LD      HL,7AB2H        ;Puffer adressieren (geles. Name)
3647  11 9D 7A      LD      DE,7A9DH        ;eingegebenen Namen adressieren
364A  1A            LD      A,(DE)          ;Byte des eingegebenen Namens laden
364B  B7            OR      A               ;Ende ?
364C  C8            RET     Z               ;ja, gefunden!
364D  BE            CP      (HL)            ;= Zeichen im Puffer ?
364E  C2 E7 35      JP      NZ,35E7H        ;nein, weiter suchen
3651  23            INC     HL              ;Pufferadresse + 1
3652  13            INC     DE              ;Adr. d. eingegeb. Namens + 1
3653  18 F5         JR      364AH           ;nächstes Zeichen
3655  C9            RET                     ;nicht benutzt

                    ************************************

                    CLOAD - Anweisung
                    Programm von Kassette lesen
3656  E5            PUSH    HL              ;Programmzeiger auf den Stack
3657  21 39 78      LD      HL,7839H        ;Flag 2 adressieren
365A  CB B6         RES     6,(HL)          ;CRUN-Flag löschen
365C  CB 9E         RES     3,(HL)          ;VERIFY-Flag löschen
365E  E1            POP     HL              ;Programmzeiger laden
```

```
                        Gemeinsam von CLOAD, CRUN und VERIFY benutzt
365F  F3                DI              ;Interrupts ausschalten
3660  CD 8C 35          CALL  358CH     ;Name aus Programmtext übernehmen
3663  E5                PUSH  HL        ;Programmzeiger auf den Stack
3664  CD B1 35          CALL  35B1H     ;Meldungsausgabe vorbereiten
3667  21 42 38          LD    HL,3842H  ;Text "WAITING" adressieren
366A  CD F4 37          CALL  37F4H     ;und ausgeben
366D  CD E7 35          CALL  35E7H     ;Programm auf Kassette suchen
3670  3A D2 7A          LD    A,(7AD2H) ;Programmkennung laden
3673  FE F2            CP    0F2H      ;sind es einfache Daten ?
3675  28 F6            JR    Z,366DH   ;ja, weiter suchen!
3677  21 60 38          LD    HL,3860H  ;Text "LOADING" adressieren
367A  CD 04 38          CALL  3804H     ;und ausgeben
367D  DD 21 23 78       LD    IX,7823H  ;Prüfsummen-Bytes adressieren
3681  CD 68 38          CALL  3868H     ;Start- und Endadresse lesen
3684  DA 11 37          JP    C,3711H   ;Fehler? ja-LOADING ERROR ausgeben
3687  E5                PUSH  HL        ;Endadresse auf den Stack
3688  ED 52             SBC   HL,DE     ;End- - Startadrresse = Bytezähler
368A  DA 11 37          JP    C,3711H   ;Start- > Endadresse? ja, Fehler
368D  ED 53 1E 78       LD    (781EH),DE ;Startadresse speichern
3691  E5                PUSH  HL        ;Bytezähler in BC übertragen
3692  C1                POP   BC
3693  E1                POP   HL        ;Endadresse laden
3694  3A 39 78          LD    A,(7839H) ;Flag 2 laden
3697  CB 5F             BIT   3,A       ;VERIFY-Flag gesetzt ?
3699  C2 42 37          JP    NZ,3742H  ;ja, zur VERIFY - Routine
369C  CD 73 3F          CALL  3F73H     ;Byte von Kassette lesen
369F  12                LD    (DE),A    ;und im RAM-Bereich speichern
36A0  CD 8E 38          CALL  388EH     ;auf Prüfsumme addieren
36A3  13                INC   DE        ;Programmadresse + 1
36A4  0B                DEC   BC        ;Bytezähler - 1
36A5  79                LD    A,C       ;= 0 ?
36A6  B0                OR    B
36A7  20 F3             JR    NZ,369CH  ;nein, weiter lesen
36A9  CD 75 37          CALL  3775H     ;LSB Prüfsumme lesen
36AC  DD BE 00          CP    (IX)      ;mit errechneter Prüfsumme vergl.
36AF  C2 11 37          JP    NZ,3711H  ;ungleich, LOADING ERROR ausgeben
36B2  CD 75 37          CALL  3775H     ;MSB Prüfsumme lesen
6B5   DD BE 01          CP    (IX+1)    ;mit errechneter Prüfsumme vergl.
36B8  C2 11 37          JP    NZ,3711H  ;ungleich, LOADING ERROR ausgeben
36BB  22 F9 78          LD    (78F9H),HL ;Programmendadresse speichern
36BE  FB                EI              ;Interrupts wieder einschalten
36BF  3E 0D             LD    A,0DH     ;CR - LF ausgeben
```

```
36C1  CD 8B 30      CALL  308BH
36C4  3A D2 7A      LD    A,(7AD2H)      ;Programm-Kennung laden
36C7  FE F1         CP    0F1H           ;ein Maschinenprogramm ?
36C9  20 04         JR    NZ,36CFH       ;nein!
                    Maschinenprogramm starten
36CB  2A 1E 78      LD    HL,(781EH)     ;Startadresse laden
36CE  E9            JP    (HL)           ;und anspringen
                    BASIC - Programm
36CF  21 29 19      LD    HL,1929H       ;Text "READY" adressieren
36D2  CD A7 28      CALL  28A7H          ;und ausgeben
36D5  2A A4 78      LD    HL,(78A4H)     ;Programm-Startadresse laden
36D8  E5            PUSH  HL             ;und auf den Stack
36D9  21 39 78      LD    HL,7839H       ;Flag 2 adressieren
36DC  CB 76         BIT   6,(HL)         ;CRUN-Flag gesetzt ?
36DE  20 03         JR    NZ,36E3H       ;ja, anstarten
36E0  C3 E8 1A      JP    1AE8HH         ;nein, zur Hauptschleife
                    BASIC-Programm starten
36E3  21 39 78      LD    HL,7839H       ;Flag 2 adressieren
36E6  CB B6         RES   6,(HL)         ;CRUN-Flag löschen
36E8  D1            POP   DE             ;Programm-Startadresse laden
36E9  CD FC 1A      CALL  1AFCH          ;Zeilenzeiger erneuern
36EC  CD B5 79      CALL  79B5H          ;RAM-Erweiterungsausgang
36EF  CD 5D 1B      CALL  1B5DH          ;Variablen-Tabelle löschen
36F2  CD B8 78      CALL  78B8H          ;RAM-Erweiterungsausgang
36F5  21 FF FF      LD    HL,0FFFFH      ;akt. Zeilennummer = Direktbefehl
36F8  22 A2 78      LD    (78A2H),HL
36FB  21 E8 79      LD    HL,79E8H       ;I/O-Puffer adressieren
36FE  11 70 05      LD    DE,0570H       ;RUN-Befehl adressieren
3701  1A            LD    A,(DE)         ;und in I/O-Puffer übertragen
3702  77            LD    (HL),A
3703  B7            OR    A              ;Ende ?
3704  28 04         JR    Z,370AH        ;ja, fertig
3706  23            INC   HL             ;Pufferadresse + 1
3707  13            INC   DE             ;Textadresse + 1
3708  18 F7         JR    3701H          ;nächstes Zeichen
370A  21 E7 79      LD    HL,79E7H       ;I/O-Puffer - 1 adressieren
370D  AF            OR    A
370E  C3 81 1A      JP    1A81H          ;RUN-Befehl ausführen

                    Meldung über Ladefehler ausgeben
3711  21 4A 38      LD    HL,384AH       ;Text 'LOADING ERROR" adressieren
3714  FB            EI                   ;Interrupts einschalten
3715  CD A7 28      CALL  28A7H          ;Text ausgeben
3718  F3            DI                   ;Interrupts einschalten
```

```
3719  3A 4C 78        LD      A,(784CH)       ;Ausgabe-Flag laden
371C  B7              OR      A               ;= 0 ?
371D  C2 67 36        JP      NZ,3667H        ;nein, weitere Textausgabe unterdr.
3720  21 FF 71        LD      HL,71FFH        ;Cursor auf letzte Zeichenstelle
3723  22 20 78        LD      (7820H),HL
3726  3E 1F           LD      A,31            ;Spaltenzeiger auf letzte Spalte
3728  32 A6 78        LD      (78A6H),A
372B  C3 67 36        JP      3667H           ;noch einmal versuchen

                      ************************************

                      CRUN - Anweisung
                      einlesen und starten eines Programms
372E  E5              PUSH    HL              ;Programmzeiger auf den Stack
372F  21 39 78        LD      HL,7839H        ;Flag 2 adressieren
3732  CB F6           SET     6,(HL)          ;CRUN-Flag setzen
3734  E1              POP     HL              ;Programmzeiger laden
3735  C3 5F 36        JP      365FH

                      ************************************

                      VERIFY - Anweisung
                      prüfen eines Programms auf Kassette
3738  E5              PUSH    HL              ;Programmzeiger auf den Stack
3739  21 39 78        LD      HL,7839H        ;Flag 2 adressieren
373C  CB DE           SET     3,(HL)          ;VERIFY-Flag setzen
373E  E1              POP     HL              ;Programmzeiger laden
373F  C3 5F 36        JP      365FH

                      ************************************

                      VERIFY - Forsetzung nach gemeinsamer Routine mit CLOAD
3742  EB              EX      DE,HL           ;HL = Programm-Startadresse
3743  CD 75 37        CALL    3775H           ;Byte von Kassette lesen
3746  BE              CP      (HL)            ;= Programmbyte ?
3747  28 09           JR      Z,3752H         ;ja!
3749  21 6C 37        LD      HL,376CH        ;Text "VERIFY " adressieren
374C  CD A7 28        CALL    28A7H           ;und ausgeben
374F  C3 83 01        JP      0183H           ;Text "ERROR" ausgeben

3752  23              INC     HL              ;Programmadresse + 1
3753  0B              DEC     BC              ;Bytezähler - 1
3754  79              LD      A,C             ;fertig ?
3755  B0              OR      B
```

```
3756  20 EB          JR    NZ,3743H      ;nein, nächstes Byte
3758  21 39 78       LD    HL,7839H      ;Flag 2 adressieren
375B  CB 9E          RES   3,(HL)        ;VERIFY-Flag löschen
375D  21 6C 37       LD    HL,376CH      ;Text "VERIFY " adressieren
3760  CD A7 28       CALL  28A7H         ;und ausgeben
3763  21 80 03       LD    HL,0380H      ;Text "OK" adressieren
3766  CD A7 28       CALL  28A7H         ;und ausgeben
3769  C3 CF 36       JP    36CFH         ;zurück ins BASIC

376C  0D             DEFB  0DH           ;Textdefinition "VERIFY"
376D  56 45 52 49    DEFM  'VERIFY '
      46 59 20
3774  00             DEFB  0

                     ************************************

                     Byte von Kassette lesen
                     Ausg.: A = gelesenes Byte
                            Carry gesetzt, wenn Lesefehler
3775  C5             PUSH  BC            ;Register sichern
3776  D5             PUSH  DE
3777  06 08          LD    B,8           ;Bitzähler = 8
3779  CD 8F 37       CALL  378FH         ;ein Bit lesen
377C  38 0E          JR    C,378CH       ;Lesefehler!
377E  10 F9          DJNZ  3779H         ;nächstes Bit
3780  D1             POP   DE            ;Register wieder laden
3781  C1             POP   BC
3782  32 D3 7A       LD    (7AD3H),A     ;gelesenes Byte sichern
3785  CD F8 3A       CALL  3AF8H         ;BREAK-Taste prüfen
3788  3A D3 7A       LD    A,(7AD3H)     ;gelesenes Byte wieder laden
378B  C9             RET                 ;fertig
                     Fehler-Rücksprung
378C  D1             POP   DE            ;Register wieder laden
378D  C1             POP   BC
378E  C9             RET

                     ************************************

                     Bit von Kassette lesen
378F  C5             PUSH  BC            ;BC sichern
3790  01 FF 07       LD    BC,07FFH      ;Wert für Zeitüberwachung laden
3793  3A 00 68       LD    A,(6800H)     ;I/O-Port laden
3796  CB 77          BIT   6,A           ;0 - Pegel ?
3798  28 08          JR    Z,37A2H       ;ja, weiter
```

```
379A  0B          DEC   BC              ;Zeitwert - 1
379B  79          LD    A,C             ;Zeit abgelaufen ?
379C  B0          OR    B
379D  20 F4       JR    NZ,3793H        ;nein, nochmals lesen
379F  C1          POP   BC              ;BC wieder vom Stack herstellen
37A0  37          SCF                   ;Carry-Flag setzen
37A1  C9          RET                   ;Fehler-Rücksprung
                  Prüfen, ob Clock-Puls
37A2  3A 00 68    LD    A,(6800H)       ;I/O-Port laden
37A5  CB 77       BIT   6,A             ;1 - Pegel ?
37A7  20 EA       JR    NZ,3793H        ;ja, 0-Pegel zu kurz!
37A9  3A 00 68    LD    A,(6800H)       ;I/O-Port laden
37AC  CB 77       BIT   6,A             ;1 - Pegel ?
37AE  20 E3       JR    NZ,3793H        ;ja, 0-Pegel immer noch zu kurz
37B0  06 52       LD    B,82            ;300 us Verzögerung
37B2  10 FE       DJNZ  37B2H           ;um neg.-Puls zu übergehen
37B4  3A 00 68    LD    A,(6800H)       ;I/O-Port laden
37B7  CB 77       BIT   6,A             ;jetzt 1-Pegel ?
37B9  20 09       JR    NZ,37C4H        ;ja, korrekter Clock-Puls
37BB  3A 00 68    LD    A,(6800H)       ;I/O-Port laden
37BE  CB 77       BIT   6,A             ;Pulswechsel ?
37C0  28 F9       JR    Z,37BBH         ;auf Pulsende warten
37C2  18 CC       JR    3790H           ;kein korrekter Clock-Puls
                  Anzahl Pulse messen
37C4  06 5A       LD    B,90            ;Zähler auf 1 ms setzen
37C6  0E 00       LD    C,0             ;Ergebnis-Register löschen
37C8  3A 00 68    LD    A,(6800H)       ;I/O-Port lesen
37CB  CB 77       BIT   6,A             ;auf negative Flanke warten
37CD  28 0B       JR    Z,37DAH         ;da ist sie!
37CF  10 F7       DJNZ  37C8H           ;wieder lesen
                  Anzahl Pulse - 1 = Ergebnis-Bit
37D1  79          LD    A,C             ;Anzahl Pulse in A
37D2  3D          DEC   A               ;- 1
37D3  1F          RRA                   ;niederw. Bit ins Carry
37D4  CB 12       RL    D               ;Carry-Bit in D (Byte sammeln)
37D6  C1          POP   BC              ;BC wiederherstellen
37D7  7A          LD    A,D             ;Ergebnis von D in A übertragen
37D8  B7          OR    A               ;Carry-Flag löschen
37D9  C9          RET                   ;fertig
                  Pulsstabilität prüfen und auf evtl. 2. Puls warten
37DA  3A 00 68    LD    A,(6800H)       ;I/O-Port laden
37DD  CB 77       BIT   6,A             ;wieder auf 1-Pegel?
37DF  20 EE       JR    NZ,37CFH        ;ja, zu kurz!
37E1  3A 00 68    LD    A,(6800H)       ;I/O-Port laden
```

```
37E4  CB 77        BIT   6,A          ;wieder auf 1-Pegel?
37E6  20 E7        JR    NZ,37CFH     ;ja, immer noch zu kurz!
37E8  0C           INC   C            ;Pulszähler + 1
37E9  3A 00 68     LD    A,(6800H)    ;I/O-Port laden
37EC  CB 77        BIT   6,A          ;Puls-Ende ?
37EE  20 DF        JR    NZ,37CFH     ;ja, evtl. 2. Puls erfassen
37F0  10 F7        DJNZ  37E9H        ;Zeit abgelaufen ?
37F2  18 DD        JR    37D1H        ;ja, Anzahl Pulse auswerten

                   ************************************

                   letzte Bildzeile löschen und Meldung ausgeben
37F4  3A 4C 78     LD    A,(784CH)    ;Ausgabe-Flag laden
37F7  B7           OR    A            ;= 0 ?
37F8  C0           RET   NZ           ;nein, keine Ausgabe!
37F9  11 E0 71     LD    DE,71E0H     ;letzte Zeile adressieren
37FC  06 20        LD    B,32         ;Zeilenlänge als Zähler
37FE  CD F6 3E     CALL  3EF6H        ;Leerzeichen in letzte Zeile
3801  13           INC   DE
3802  10 FA        DJNZ  37FEH
3804  3A 4C 78     LD    A,(784CH)    ;Ausgabe-Flag laden
3807  B7           OR    A            ;= 0 ?
3808  C0           RET   NZ           ;nein, keine Ausgabe!
3809  CD 0E 3F     CALL  3F0EH        ;abh.vom Hintergrund ggf.invertiert
380C  7E           LD    A,(HL)       ;Textzeichen laden
380D  B7           OR    A            ;Textende ?
380E  C8           RET   Z            ;ja, fertig!
380F  12           LD    (DE),A       ;ins Bild übertragen
3810  13           INC   DE           ;Bildadresse + 1
3811  23           INC   HL           ;Textadresse + 1
3812  18 F8        JR    380CH        ;nächstes Zeichen übertragen

                   ************************************

                   Programm-/Datei-kennung und Namen ausgeben
3814  3A 4C 78     LD    A,(784CH)    ;Ausgabe-Flag laden
3817  B7           OR    A            ;= 0 ?
3818  C0           RET   NZ           ;nein, Ausgabe unterdrücken
3819  11 E9 71     LD    DE,71E9H     ;Bildposition in letzter Zeile adr.
381C  E5           PUSH  HL           ;Adresse des Namens sichern
381D  3A D2 7A     LD    A,(7AD2H)    ;Programmkennung laden
3820  E6 0F        AND   0FH          ;oberes Halbbyte löschen
3822  21 3F 38     LD    HL,383FH     ;Kennungstabelle adressieren
3825  85           ADD   A,L          ;+ Kennung
```

```
3826  6F              LD      L,A
3827  3E 00           LD      A,0
3829  8C              ADC     A,H
382A  67              LD      H,A
382B  CD 21 3F        CALL    3F21H           ;Kennung abh.vom Hintergrund ausg.
382E  00 00           2 x NOP
3830  12              LD      (DE),A          ;':' als Trenner ausgeben
3831  13              INC     DE              ;Bildadresse + 2
3832  13              INC     DE
3833  E1              POP     HL              ;Adresse des Namens laden
3834  7E              LD      A,(HL)          ;Zeichen aus Namen laden
3835  B7              OR      A               ;Ende ?
3836  C8              RET     Z               ;ja, fertig !
3837  CD 33 3F        CALL    3F33H           ;Zeichen abh.vom Hintergrund ausg.
383A  13              INC     DE              ;Bildadresse + 1
383B  23              INC     HL              ;Namens-Adresse + 1
383C  18 F6           JR      3834H           ;nächstes Zeichen
383E  C9              RET                     ;unbenutzt

                      Kennungs-Tabelle
383F  14 02 04        DEFB    14H,02H,04H     ;Codes für T, B, D

                      ****************************************

                      Textdefinitionen für Kassetten-Routinen

3842  57 41 49 54     DEFM    'WAITING'
      49 4E 47
3849  00              DEFB    0

384A  0D              DEFB    0DH
384B  4C 4F 41 44 49  DEFM    'LOADING ERROR'
      4E 47 20 45 52
      52 4F 52
3858  0D 00           DEFB    0DH,00H

385A  46 4F 55 4E 44  DEFM    'FOUND'
385F  00              DEFB    0

3860  4C 4F 41 44     DEFM    'LOADING'
      49 4E 47
3867  00              DEFB    0

                      ****************************************
```

```
                        Start- und Endadresse von Kassette lesen
3868  CD 75 37          CALL    3775H           ;Byte lesen
386B  D8                RET     C               ;Lesefehler!
386C  5F                LD      E,A             ;LSB Startadresse in E
386D  DD 77 00          LD      (IX),A          ;und in LSB Prüfsumme
3870  AF                XOR     A               ;MSB Prüfsumme = 0
3871  DD 77 01          LD      (IX+1),A
3874  CD 75 37          CALL    3775H           ;Byte lesen
3877  D8                RET     C               ;Lesefehler!
3878  57                LD      D,A             ;MSB Startadresse in D
3879  CD 8E 38          CALL    388EH           ;und auf Prüfsumme addieren
387C  CD 75 37          CALL    3775H           ;Byte lesen
387F  D8                RET     C               ;Lesefehler!
3880  6F                LD      L,A             ;LSB Endadresse in L
3881  CD 8E 38          CALL    388EH           ;und auf Prüfsumme addieren
3884  CD 75 37          CALL    3775H           ;Byte lesen
3887  D8                RET     C               ;Lesefehler!
3888  67                LD      H,A             ;MSB Endadresse in H
3889  CD 8E 38          CALL    388EH           ;und auf Prüfsumme addieren
388C  B7                OR      A               ;Carry-Flag löschen
388D  C9                RET                     ;fertig

                        ************************************

                        Prüfsumme ermitteln
                        Eing.: IX = Adresse der Prüfsummen-Bytes
                               A  = eingelesenes Byte
388E  DD 86 00          ADD     A,(IX)          ;LSB Prüfsumme auf Zeichen addieren
3891  DD 77 00          LD      (IX),A          ;und wieder speichern
3894  3E 00             LD      A,0             ;A = 0
3896  DD 8E 01          ADC     A,(IX+1)        ;MSB Prüfsumme auf Carry addieren
3899  DD 77 01          LD      (IX+1),A        ;und wieder speichern
389C  C9                RET                     ;fertig

                        ************************************

                        COLOR - Anweisung
389D  7E                LD      A,(HL)          ;nächstes Programmzeichen laden
389E  FE 2C             CP      ','             ;= Komma ?
38A0  28 20             JR      Z,38C2H         ;ja, nur Hintergrund ändern
38A2  CD 1C 2B          CALL    2B1CH           ;1. Ausdruck auswerten
38A5  B7                OR      A               ;= 0 ?
38A6  CA 4A 1E          JP      Z,1E4AH         ;ja, FUNCTION CODE - Error
```

```
38A9  FE 09        CP    9              ;> 8 ?
38AB  D2 4A 1E     JP    NC,1E4AH       ;ja, FUNCTION CODE - Error
38AE  3D           DEC   A              ;Farbcode - 1
38AF  E6 07        AND   7              ;auf 0 - 7 beschränken
38B1  CB 27        SLA   A              ;ins obere Halbbyte schieben
38B3  CB 27        SLA   A
38B5  CB 27        SLA   A
38B7  CB 27        SLA   A
38B9  32 46 78     LD    (7846H),A      ;und abspeichern
38BC  7E           LD    A,(HL)         ;nächstes Zeichen laden
38BD  B7           OR    A              ;Zeilenende ?
38BE  C8           RET   Z              ;ja, fertig!
38BF  FE 3A        CP    ':'            ;Kommando-Ende ?
38C1  C8           RET   Z              ;ja, fertig!
38C2  CF           RST   8              ;nächstes Zeichen Komma ?
38C3  2C           DEFB  ','
38C4  CD 1C 2B     CALL  2B1CH          ;ja, 2. Ausdruck auswerten
38C7  B7           OR    A              ;= 0 ?
38C8  20 0C        JR    NZ,38D6H       ;nein!
38CA  3A 3B 78     LD    A,(783BH)      ;I/O-Latch Byte laden
38CD  CB A7        RES   4,A            ;Hintergrund auf grün setzen
38CF  32 3B 78     LD    (783BH),A      ;wieder speichern
38D2  32 00 68     LD    (6800H),A      ;und über I/O-Port ausgeben
38D5  C9           RET                  ;fertig

38D6  FE 01        CP    1              ;= 1 ?
38D8  C2 4A 1E     JP    NZ,1E4AH       ;nein, FUNCTION CODE - Error
38DB  3A 3B 78     LD    A,(783BH)      ;I/O-Latch Byte laden
38DE  CB E7        SET   4,A            ;Hintergrund auf rot setzen
38E0  32 3B 78     LD    (783BH),A      ;wieder speichern
38E3  32 00 68     LD    (6800H),A      ;und über I/O-Port ausgeben
38E6  C9           RET                  ;fertig

                   ************************************

                   Ergänzungroutine zur POINT-Funktion
38E7  0E C0        LD    C,0C0H         ;2-Bit Maske laden
38E9  CB 09        RRC   C              ;entsprechend der Pixel-Stellung
38EB  10 FC        DJNZ  38E9H          ;im Byte nach rechts schieben
38ED  1A           LD    A,(DE)         ;Byte aus Grafikspeicher laden
38EE  A1           AND   C              ;mit Maske verknüpfen
38EF  47           LD    B,A            ;Ergebnis in B sichern
38F0  79           LD    A,C            ;Maske in A laden
38F1  CB 08        RRC   B              ;Byte und Maske soweit rechts
```

```
38F3  CB 0F        RRC   A            ;schieben, bis Maske ganz rechts
38F5  FE 03        CP    3            ;prüfen ob Maske rechts
3487  20 F8        JR    NZ,38F1H     ;weiter schieben!
38F9  78           LD    A,B          ;Farbcode aus B laden
38FA  3C           INC   A            ;+ 1 für Ergebnis (1-4)
38FB  E5           PUSH  HL           ;Programmzeiger auf den Stack
38FC  CD 8D 09     CALL  098DH        ;A in 16-Bit Integer umwandeln (X)
38FF  E1           POP   HL           ;Programmzeiger wieder laden
3900  C3 0F 39     JP    390FH        ;prüfen, ob ')' folgt

                   ************************************

                   Ergänzungs-Routine zur SET- und RESET-Anweisung
3903  47           LD    B,A          ;SET-Maske in B übertragen
3904  1A           LD    A,(DE)       ;Byte aus Bildspeicher laden
3905  A1           AND   C            ;Bits für adr. Pixel löschen
3906  12           LD    (DE),A       ;Byte zurückschreiben
3907  F1           POP   AF           ;Funktions-Flag laden
3908  B7           OR    A            ;RESET-Anweisung ?
3909  F2 0F 39     JP    P,390FH      ;ja, fertig!
390C  1A           LD    A,(DE)       ;SET, Byte wieder laden
390D  B0           OR    B            ;Bits für adr. Pixel setzen
390E  12           LD    (DE),A       ;Byte zurückschreiben
390F  CF           RST   8            ;Parameter mit ')' abgeschlossen ?
3910  29           DEFB  ')'
3911  C9           RET                ;ja, fertig

                   ************************************

                   COPY - Anweisung
3912  F3           DI                 ;Interrupts ausschalten
3913  E5           PUSH  HL           ;Programmzeiger auf den Stack
3914  3A 3B 78     LD    A,(783BH)    ;I/O-Latch Byte laden
3917  CB 5F        BIT   3,A          ;Rechner im Grafik-Modus ?
3919  C2 8E 39     JP    NZ,398EH     ;ja, Grafikausgabe
                   Bildschirmausgabe im Textmodus
391C  21 00 70     LD    HL,7000H     ;Bildanfangsadresse
391F  0E 10        LD    C,16         ;Zeilenzähler = 16
3921  06 20        LD    B,32         ;Spaltenzähler = 32
3923  7E           LD    A,(HL)       ;Zeichen aus Bildspeicher laden
3924  B7           OR    A            ;Blockgrafik - Zeichen?
3925  F2 2D 39     JP    P,392DH      ;nein!
3928  CD 73 2C     CALL  2C73H        ;Ausgabe von Blockgrafik
392B  18 16        JR    3943H
```

```
392D  C3 44 3F      JP    3F44H       ;Prüfen, ob Zeichen invertiert ist
3930  00            NOP               ;wenn nein, Fortsetzung bei 3938H
3931  E6 3F         AND   3FH         ;Bits 6 und 7 löschen
3933  CD 56 39      CALL  3956H       ;Ausgabe invertierter Zeichen
3936  18 0B         JR    3943H
3938  E6 3F         AND   3FH         ;Bits 6 und 7 löschen
393A  CB 6F         BIT   5,A         ;Zeichen < 1FH ?
393C  20 02         JR    NZ,3940H    ;nein!
393E  F6 40         OR    40H         ;ja, + 40H für echtes ASCII-Zeichen
3940  CD BA 3A      CALL  3ABAH       ;Zeichen auf Drucker ausgeben
3943  23            INC   HL          ;Bildadresse + 1
3944  10 DD         DJNZ  3923H       ;Zeilenende ?
3946  3E 0D         LD    A,0DH       ;ja, CR/LF ausgeben
3948  CD BA 3A      CALL  3ABAH
394B  CD F8 3A      CALL  3AF8H       ;BREAK-Taste betätigt ?
394E  0D            DEC   C           ;Zeilenzähler - 1 ?
394F  79            LD    A,C         ;= 0 ?
3950  B7            OR    A
3951  20 CE         JR    NZ,3921H    ;nein, nächste Zeile
3953  E1            POP   HL          ;Programmzeiger laden
3954  FB            EI                ;Interrupts einschalten
3955  C9            RET               ;fertig
                    Invertierte Zeichen ausgeben
3956  F5            PUSH  AF          ;Register auf Stack sichern
3957  C5            PUSH  BC
3958  D5            PUSH  DE
3959  E5            PUSH  HL
395A  6F            LD    L,A         ;auszugebendes Zeichen in HL
395B  26 00         LD    H,0
395D  3E 08         LD    A,8         ;Drucker in Grafik-Modus schalten
395F  CD BA 3A      CALL  3ABAH
3962  06 04         LD    B,4         ;Zeichencode * 5
3964  E5            PUSH  HL          ;als Tabellen-Offset
3965  D1            POP   DE
3966  B7            OR    A
3967  ED 5A         ADC   HL,DE
3969  10 FC         DJNZ  3967H
396B  E5            PUSH  HL          ;Tabellen-Offset in BC
396C  C1            POP   BC
396D  21 94 3B      LD    HL,3B94H    ;Tabelle f. invertierte Zeichen
3970  09            ADD   HL,BC       ;+ Offset
3971  3E FF         LD    A,0FFH      ;1 Reihe Punkte links des Zeichens
3973  CD BA 3A      CALL  3ABAH       ;ausgeben
3976  06 05         LD    B,5         ;5 Punktreihen aus Tabelle
```

```
3978  7E           LD    A,(HL)          ;ausgeben
3979  23           INC   HL
397A  CD BA 3A     CALL  3ABAH
397D  10 F9        DJNZ  3978H
397F  3E FF        LD    A,0FFH          ;1 Reihe Punkte rechts des
3981  CD BA 3A     CALL  3ABAH           ;Zeichens ausgeben
3984  3E 0F        LD    A,0FH           ;Drucker wieder in Text-Modus
3986  CD BA 3A     CALL  3ABAH           ;schalten
3989  E1           POP   HL              ;Registerinhalte wiederherstellen
398A  D1           POP   DE
398B  C1           POP   BC
398C  F1           POP   AF
398D  C9           RET                   ;fertig!
                   Bild im Grafik-Modus ausdrucken
398E  AF           XOR   A               ;Intervallzähler rücksetzen
398F  32 D6 7A     LD    (7AD6H),A
3992  32 D6 7A     LD    (7AD6H),A
3995  3E 08        LD    A,8             ;Drucker in Grafik-Modus
3997  CD BA 3A     CALL  3ABAH           ;schalten
399A  DD 21 D2 7A  LD    IX,(7AD2H)      ;Anf.adresse Grafik-Puffer laden
399E  21 00 70     LD    HL,7000H        ;Bildanfangs-Adresse laden
39A1  11 00 00     LD    DE,0            ;Druck-Pattern rücksetzen
39A4  0E C0        LD    C,0C0H          ;Bits 6,7 in Shift-Maske setzen
39A6  CD F8 3A     CALL  3AF8H           ;BREAK-Taste gedrückt ?
39A9  E5           PUSH  HL              ;Bildadresse auf den Stack
39AA  CD C9 05     CALL  05C9H           ;Grafik-Puffer löschen
                   jeweils 3 Zeilen in einem Grafikzeichen kombinieren
39AD  06 03        LD    B,3             ;Zeilenzähler = 3
39AF  7E           LD    A,(HL)          ;Grafik-Byte aus Bildspeicher laden
39B0  A1           AND   C               ;mit Shift-Maske 1 Pixel auswählen
39B1  C5           PUSH  BC              ;Zähler + Shift-maske auf Stack
39B2  47           LD    B,A             ;das ausgewählte Pixel in B
39B3  CB 08        RRC   B               ;mithilfe der SHIFT-Maske so oft
39B5  CB 08        RRC   B               ;rechts rotieren, bis die zwei
39B7  CB 09        RRC   C               ;Pixel-Bits in den Bitpositionen
39B9  CB 09        RRC   C               ;0 und 1 stehen
39BB  79           LD    A,C
39BC  FE 03        CP    3               ;Maske rechtsbündig ?
39BE  C2 B3 39     JP    NZ,39B3H        ;nein, weiter schieben
39C1  78           LD    A,B             ;Pixel wieder in A
39C2  C1           POP   BC              ;Zähler + Shift-Maske wieder laden
39C3  FE 03        CP    3               ;Pixel-Farbe = rot ?
39C5  28 0D        JR    Z,39D4H         ;ja!
39C7  FE 02        CP    2               ;Pixel-Farbe = blau ?
```

```
39C9  28 0E        JR    Z,39D9H       ;ja!
39CB  FE 01        CP    1             ;Pixel-Farbe = gelb ?
39CD  28 10        JR    Z,39DFH       ;ja!
39CF  11 00 00     LD    DE,0          ;grün - Druck-Pattern = 0000 setzen
39D2  18 0F        JR    39E3H
39D4  11 E0 E0     LD    DE,0E0E0H     ;rot - Druck-Pattern = E0E0 setzen
39D7  18 0A        JR    39E3H
39D9  16 40        LD    D,40H         ;blau - Druck-Pattern = 40A0 setzen
39DB  1E A0        LD    E,0A0H
39DD  18 04        JR    39E3H
39DF  16 A0        LD    D,0A0H        ;gelb - Druck-Pattern = 0A40 setzen
39E1  1E 40        LD    E,40H
                   Druck-Pattern aus drei Zeilen kombinieren
39E3  DD 7E 00     LD    A,(IX)        ;1. Byte aus Puffer laden
39E6  CB 3F        SRL   A             ;Bits vorherg. Zeile 3 Bitpos.
39E8  CB 3F        SRL   A             ;rechts schieben
39EA  CB 3F        SRL   A
39EC  E5           PUSH  HL            ;Bildadresse' sichern
39ED  21 D3 7A     LD    HL,7AD3H      ;Grafik-Puffer + 1
39F0  CD 6A 3A     CALL  3A6AH         ;wenn Carry, in Pufferadr.+1 übern.
39F3  E1           POP   HL            ;Bildadresse' laden
39F4  B2           OR    D             ;Byte 1 der Druck-Pattern in Puffer
39F5  DD 77 00     LD    (IX),A        ;1. Puffer-Byte zurückschreiben
39F8  DD 7E 02     LD    A,(IX+2)      ;3. Byte aus Puffer laden
39FB  CB 3F        SRL   A             ;Bits vorherg. Zeile 3 Bitpos.
39FD  CB 3F        SRL   A             ;rechts schieben
39FF  CB 3F        SRL   A
3A01  E5           PUSH  HL            ;Bildadresse' sichern
3A02  21 D5 7A     LD    HL,7AD5H      ;Grafik-Puffer + 3
3A05  CD 6A 3A     CALL  3A6AH         ;wenn Carry, in Pufferadr.+3 übern.
3A08  E1           POP   HL            ;Bildadresse' laden
3A09  B3           OR    E             ;Byte 2 aus Druck-Pattern in Puffer
3A0A  DD 77 02     LD    (IX+2),A      ;3. Puffer-Byte zurückschreiben
3A0D  3E 20        LD    A,32          ;Bildadresse' + 1 Zeile
3A0F  85           ADD   A,L
3A10  6F           LD    L,A
3A11  3E 00        LD    A,0
3A13  8C           ADC   A,H
3A14  67           LD    H,A
3A15  10 50        DJNZ  3A67H         ;3 Zeilen bearbeitet ?
3A17  CD 73 3A     CALL  3A73H         ;ja, ausdrucken!
                   die nächsten 3 Pixel in derselben Zeile
3A1A  E1           POP   HL            ;Bildadresse laden
3A1B  CB 39        SRL   C             ;Shift-Maske 2 Bits
```

```
3A1D  CB 39      SRL   C              ;rechts schieben
3A1F  79         LD    A,C            ;3-Byte-Reihe fertig bearbeitet ?
3A20  B7         OR    A
3A21  20 83      JR    NZ,39A6H       ;nein, nächste Reihe im selben Byte
3A23  23         INC   HL             ;ja, Bildadresse + 1
3A24  7D         LD    A,L            ;Ende der Zeile ?
3A25  E6 1F      AND   1FH
3A27  C2 A4 39   JP    NZ,39A4H       ;nein, nächstes Byte
3A2A  CD E2 3A   CALL  3AE2H          ;CR/LF für neue Zeile ausgeben
                 nächste Zeile adressieren
3A2D  3A D6 7A   LD    A,(7AD6H)      ;Intervallzähler laden
3A30  3C         INC   A              ;+ 1
3A31  FE 03      CP    3              ;4. Zeile ?
3A33  20 01      JR    NZ,3A36H       ;nein!
3A35  AF         XOR   A              ;Intervallzähler = 0
3A36  32 D6 7A   LD    (7AD6H),A      ;neuen Wert in Intervallzähler
3A39  20 04      JR    NZ,3A3FH
3A3B  3E 40      LD    A,64           ;A = Länge von zwei Zeilen
3A3D  18 02      JR    3A41H
3A3F  3E 20      LD    A,32           ;A = Länge einer Zeile
3A41  85         ADD   A,L            ;HL + eine o. zwei Zeilen
3A42  6F         LD    L,A
3A43  3E 00      LD    A,0
3A45  8C         ADC   A,H
3A46  67         LD    H,A
3A47  FE 78      CP    78H            ;außerhalb des Bildes ?
3A49  D2 5F 3A   JP    NC,3A5FH       ;ja, fertig!
3A4C  FE 77      CP    77H            ;letzte Zeile ?
3A4E  C2 A4 39   JP    NZ,39A4H       ;nein!
3A51  7D         LD    A,L
3A52  FE E0      CP    0E0H
3A54  DA A4 39   JP    C,39A4H        ;nein!
3A57  3E FF      LD    A,0FFH         ;Kenner 'letzte Zeile' setzen
3A59  32 D6 7A   LD    (7AD6H),A
3A5C  C3 A4 39   JP    39A4H          ;nächste Zeile ausdrucken
                 Grafik-Ausdruck erstellt!
3A5F  3E 0F      LD    A,0FH          ;Drucker in Text-Modus schalten
3A61  CD BA 3A   CALL  3ABAH
3A64  E1         POP   HL             ;Programmzeiger laden
3A65  FB         EI                   ;Interrupts einschalten
3A66  C9         RET                  ;fertig!
3A67  C3 AF 39   JP    39AFH          ;was soll das ?
                 bei SHIFT aufgetretenes Carry ins nächste Puffer-Byte
3A6A  D2 70 3A   JP    NC,3A70H       ;kein Carry!
```

```
3A6D  CB C6       SET   0,(HL)        ;Carry, Bit 0 im n. Puffer-Byte = 1
3A6F  C9          RET
3A70  CB 86       RES   0,HL)         ;Bit 0 im nächsten Puffer-Byte = 0
3A72  C9          RET
                  Grafik-Puffer ausgeben
3A73  CD 85 3A    CALL  3A85H         ;Puffer 1+2 ausgeben
3A76  DD 23       INC   IX
3A78  DD 23       INC   IX
3A7A  CD 85 3A    CALL  3A85H         ;Puffer 3+4 ausgeben
3A7D  DD 2B       DEC   IX
3A7F  DD 2B       DEC   IX
3A81  CD 85 3A    CALL  3A85H         ;Puffer 1+2 ausgeben
3A84  C9          RET
                  eine Punktreihe für Grafik-Druck ausgeben
3A85  DD 7E 01    LD    A,(IX+1)      ;übertrag aus Puffer+1 o. 3 laden
3A88  CB 0F       RRC   A             ;ins Carry schieben
3A8A  DD 7E 00    LD    A,(IX)        ;Puffer + 0 oder 2 laden
3A8D  F5          PUSH  AF            ;auf Stack merken
3A8E  3A D6 7A    LD    A,(7AD6H)     ;Intervall-Zähler laden
3A91  FE 02       CP    2             ;3. Zeile ?
3A93  28 1D       JR    Z,3AB2H       ;ja!
3A95  FE 01       CP    1             ;2. Zeile ?
3A97  28 16       JR    Z,3AAFH       ;ja!
3A99  F1          POP   AF            ;Puffer + 1 oder 2 laden
3A9A  17          RLA                 ;Carry-Bit in niederw. Bitposition
3A9B  F5          PUSH  AF            ;auszugebendes Zeichen merken
3A9C  3A D6 7A    LD    A,(7AD6H)     ;Intervall-Zähler laden
3A9F  FE FF       CP    0FFH          ;letzte Bildzeile ?
3AA1  20 05       JR    NZ,3AA8H      ;nein!
3AA3  F1          POP   AF            ;auszug. Zeichen wieder laden
3AA4  E6 07       AND   7             ;nur die unteren 3 Bits ausgeben
3AA6  18 01       JR    3AA9H
3AA8  F1          POP   AF            ;auszug. Zeichen wieder laden
3AA9  F6 80       OR    80H           ;Bit 7 setzen
3AAB  CD BA 3A    CALL  3ABAH         ;Grafik-Zeichen ausgeben
3AAE  C9          RET                 ;fertig!

3AAF  F1          POP   AF            ;auszugebendes Zeichen laden
3AB0  18 E9       JR    3A9BH         ;unverändert ausgeben

3AB2  F1          POP   AF            ;auszugebendes Zeichen laden
3AB3  1F          RRA                 ;übertrag in höchstw. Bitposition
3AB4  18 E5       JR    3A9BH         ;und ausgeben
```

```
                        ************************************

                        Zeichen auf dem Drucker ausgeben
3AB6  B7                OR      A               ;Blockgrafik oder invertiert ?
3AB7  FA DE 3A          JP      M,3AD8H         ;ja!
3ABA  F5                PUSH    AF              ;auszug. Zeichen sichern
3ABB  CD E8 3A          CALL    3AE8H           ;BREAK-Taste gedrückt ?
3ABE  D2 C4 3A          JP      NC,33AC4H       ;nein!
3AC1  F1                POP     AF              ;auszug. Zeichen laden
3AC2  37                SCF                     ;Carry-Flag setzen
3AC3  C9                RET                     ;und zurück
3AC4  DB 00             IN      A,(00)          ;Port 0 lesen
3AC6  CB 47             BIT     0,A             ;BUSY ?
3AC8  20 F1             JR      NZ,3ABBH        ;ja, warten
3ACA  F1                POP     AF              ;auszug. Zeichen laden
3ACB  D3 0E             OUT     (0EH),A         ;Byte ausgeben
3ACD  D3 0D             OUT     (0DH),A         ;Strobe ausgeben
3ACF  FE 0D             CP      0DH             ;Carriage Return ?
3AD1  37                SCF                     ;Set Carry
3AD2  3F                CCF                     ;Reset Carry
3AD3  C0                RET     NZ              ;nein!
3AD4  3E 0A             LD      A,0AH           ;Line-Feed ausgeben
3AD6  18 E2             JR      3ABAH
3AD8  CB 77             BIT     6,A             ;invertiertes Zeichen ?
3ADA  CA 73 2C          JP      Z,2C73H         ;nein, Blockgrafik ausgeben
3ADD  E6 3F             AND     3FH             ;Bits 6,7 löschen
3ADF  C3 56 39          JP      3956H           ;invertiertes Zeichen ausgeben

                        ************************************

                        Carriage-Return ausgeben (vom Treiber 5A6)
3AE2  3E 0D             LD      A,0DH           ;Code für CR laden
3AE4  CD BA 3A          CALL    3ABAH           ;und asgeben
3AE7  C9                RET

                        ************************************

                        Prüfen, ob BREAK-Taste betätigt
                        Ausg.: Carry=1, wenn gedrückt
3AE8  B7                OR      A               ;Carry löschen
3AE9  3A FD 68          LD      A,(68FDH)       ;Tastaturzeile 2 laden
3AEC  CB 57             BIT     2,A             ;CTRL-Taste ?
3AEE  C0                RET     NZ              ;nein!
3AEF  3A DF 68          LD      A,(68DFH)       ;Tastaturzeile 6 laden
```

```
3AF2  37          SCF                 ;Carry-Flag setzen
3AF3  CB 57       BIT   2,A           ;BREAK-Taste ?
3AF5  C8          RET   Z             ;ja!
3AF6  3F          CCF                 ;nein, Carry-Flag löschen
3AF7  C9          RET

                  ************************************

                  BREAK-Taste prüfen
                  wenn gedrückt, Programmausführung unterbrechen
3AF8  CD E8 3A    CALL  3AE8H         ;BREAK-Taste prüfen
3AFB  D0          RET   NC            ;nicht gedrückt!
3AFC  E1          POP   HL            ;Rücksprungadresse entfernen
3AFD  E1          POP   HL            ;Programmzeiger laden
3AFE  3A 39 78    LD    A,(7839H)     ;Flag 2 laden
3B01  E6 B7       AND   0B7H          ;CRUN- u. VERIFY-Flag löschen
3B03  32 39 78    LD    (7839H),A     ;Flag 2 speichern
3B06  3E 01       LD    A,1
3B08  FB          EI                  ;Interrupts einschalten
3B09  C3 A0 1D    JP    1DA0H         ;zurück zum BASIC

                  ************************************

                  Bei Bildschirmausgabe warten, bis Ausgabepuffer
                  vollständig ausgegeben wurde
3B0C  3A 9C 78    LD    A,(789CH)     ;Geräte-Typ laden
3B0F  B7          OR    A             ;= 0 ? (Bildschirm)
3B10  C2 64 21    JP    NZ,2164H      ;nein, sofort zurück
3B13  3A AF 7A    LD    A,(7AAFH)     ;Pufferzähler laden
3B16  B7          OR    A             ;= 0 ? (Puffer leer)
3B17  20 FA       JR    NZ,3B13H      ;nein, warten
3B19  C3 64 21    JP    2164H         ;ja, zurück

                  ************************************

                  wenn Bildausgabe-Puffer leer, Spaltenzeiger laden
3B1C  3A AF 7A    LD    A,(7AAFH)     ;Pufferzähler laden
3B1F  B7          OR    A             ;Puffer leer ?
3B20  C0          RET   NZ            ;nein, zurück
3B21  3A A6 78    LD    A,(78A6H)     ;ja, Spaltenzeiger laden
3B24  C9          RET

                  ************************************
```

```
                        Ausgabe-Unterbrechung bei List-Ausgabe
3B25  21 EF 68          LD      HL,68EFH        ;Tastaturzeile 4 adressieren
3B28  CB 66             BIT     4,(HL)          ;Leertaste betätigt ?
3B2A  20 18             JR      NZ,3B44H        ;nein!
3B2C  CD 48 3B          CALL    3B48H           ;Taste entprellen
3B2F  CB 66             BIT     4,(HL)          ;warten, bis Taste losgelassen wird
3B31  28 FC             JR      Z,3B2FH
3B33  CD 48 3B          CALL    3B48H           ;Taste entprellen
3B36  CD F8 3A          CALL    3AF8H           ;BREAK-Taste gedrückt ?
3B39  CB 66             BIT     4,(HL)          ;nein, Leertaste erneut betätigt ?
3B3B  20 F9             JR      NZ,3B36H        ;nein, warten
3B3D  CD 48 3B          CALL    3B48H           ;Taste entprellen
3B40  CB 66             BIT     4,(HL)          ;warten, bis Taste losgelassen wird
3B42  28 FC             JR      Z,3B40H
3B44  21 FF FF          LD      HL,0FFFFH       ;akt.Zeilennummer = Direktkommando
3B47  C9                RET
                        Taste entprellen
3B48  21 FF 07          LD      HL,7FFH         ;Zähler für Warteschleife
3B4B  2B                DEC     HL              ;Zähler - 1
3B4C  7D                LD      A,L             ;= 0 ?
3B4D  B4                OR      H
3B4E  20 FB             JR      NZ,3B4BH        ;nein, weiter zählen
3B50  21 EF 68          LD      HL,68EFH        ;Tastaturzeile 4 adressieren
3B53  C9                RET                     ;fertig!

                        ************************************

                        bei PRINT# Daten auf Kassette ausgeben
3B54  CD 11 35          CALL    3511H           ;Byte auf Kassette schreiben
3B57  C9                RET

                        ************************************

                        bei PRINT# Vorspann auf Kassette schreiben
3B58  F3                DI                      ;Interrupts ausschalten
3B59  23                INC     HL              ;Programmzeiger + 1
3B5A  0E F2             LD      C,0F2H          ;Kenner für Daten laden
3B5C  CD 58 35          CALL    3558H           ;Sync-Bytes, Vorspann, Kenner und
                                                ;Dateinamen auf Kassette ausgeben
3B5F  DA FE 3A          JP      C,3AFEH         ;BREAK! Ende
3B62  2B                DEC     HL              ;Programmzeiger - 1
3B63  CF                RST     8               ;ist Dateiname mit '"' abgeschl. ?
3B64  22                DEFB    '"'
3B65  CF                RST     8               ;folgt dann ein Komma ?
```

```
3B66  2C              DEFB   ','
3B67  C9                                    ;ja, fertig!

                      ************************************

                      bei INPUT# Vorspann und Dateinamen von Kassette lesen
3B68  F3              DI                    ;Interrupts ausschalten
3B69  23              INC    HL             ;Programmzeiger + 1
3B6A  CD 8C 35        CALL   358CH          ;Dateinamen übernehmen
3B6D  2B              DEC    HL             ;Programmzeiger - 1
3B6E  CF              RST    8              ;Name mit '"' abgeschlossen ?
3B6F  22              DEFB   '"'
3B70  CF              RST    8              ;folgt dann ein Komma ?
3B71  2C              DEFB   ','
3B72  E5              PUSH   HL             ;ja, Programmzeiger auf den Stack
3B73  CD B1 35        CALL   35B1H          ;Meldungsausgabe vorbereiten
3B76  21 42 38        LD     HL,3842H       ;Text "WAITING" adressieren
3B79  CD F4 37        CALL   37F4H          ;und ausgeben
3B7C  CD E7 35        CALL   35E7H          ;Datei auf Kassette suchen
3B7F  3A D2 7A        LD     A,(7AD2H)      ;Datei-Kenner laden
3B82  FE F2           CP     [illegible]2H  ;korrekt für Daten ?
3B84  20 F6           JR     NZ,3B7CH       ;nein, weiter suchen
3B86  E1              POP    HL             ;ja, Programmzeiger laden
3B87  C9              RET                   ;fertig!

                      ************************************

                      Bei INPUT# Daten von Kassette lesen
3B88  CD 75 337       CALL   3775H          ;Byte von Kassette lesen
3B8B  FE 0D           CP     0DH            ;Satzende ?
3B8D  C0              RET    NZ             ;nein, zurück
3B8E  F5              PUSH   AF             ;Zeichen auf Stack sichern
3B8F  CD F9 20        CALL   20F9H          ;CR/LF ausgeben
3B92  F1              POP    AF             ;Zeichen wieder laden
3B93  C9              RET                   ;fertig!

                      ************************************

                      Pixel-Tabelle für inverse Zeichenausgabe auf
                      dem Drucker. Pro Zeichen 5 Byte
3B94  C1 BE A2 AE B1  DEFB   0C1H,0BEH,0A2H,0AEH,0B1H     ;0
3B99  83 ED EE ED 83  DEFB   83H,0EDH,0EEH,0EDH,83H       ;A
3B9E  80 B6 B6 B6 C1  DEFB   80H,0B6H,0B6H,0B6H,0C1H      ;B
3BA3  C1 BE BE BE DD  DEFB   0C1H,0BEH,0BEH,0BEH,0DDH     ;C
```

```
3BA8  80 BE BE BE C1    DEFB   80H,BEH,0BEH,0BEH,0C1H      ;D
3BAD  80 B6 B6 B6 BE    DEFB   80H,0B6H,0B6H,0B6H,0BEH     ;E
3BB2  80 F6 F6 F6 FE    DEFB   80H,0F6H,0F6H,0F6H,0FEH     ;F
3BB7  C1 BE BE AE 8C    DEFB   0C1H,BEH,0BEH,0AEH,8CH      ;G
3BBC  80 F7 F7 F7 80    DEFB   80H,0F7H,0F7H,0F7H,80H      ;H
3BC1  FF BE 80 BE FF    DEFB   0FFH,0BEH,80H,0BEH,0FFH     ;I
3BC6  DF BF BF C0 FE    DEFB   0DFH,0BFH,0BFH,0C0H,0FEH    ;J
3BCB  80 F7 EB DD BE    DEFB   80H,0F7H,0EBH,0DDH,0BEH     ;K
3BD0  80 BF BF BF BF    DEFB   80H,0BFH,0BFH,0BFH,0BFH     ;L
3BD5  80 FD F3 FD 80    DEFB   80H,0FDH,0F3H,0FDH,80H      ;M
3BDA  80 FD FB F7 80    DEFB   80H,0FDH,0FBH,0F7H,80H      ;N
3BDF  C1 BE BE BE C1    DEFB   0C1H,0BEH,0BEH,0BEH,0C1H    ;O
3BE4  80 F6 F6 F6 F9    DEFB   80H,0F6H,0F6H,0F6H,0F9H     ;P
3BE9  C1 BE AE DE A1    DEFB   0C1H,0BEH,0AEH,0DEH,0A1H    ;Q
3BEE  80 F6 E6 D6 B9    DEFB   80H,0F6H,0E6H,0D6H,0B9H     ;R
3BF3  D9 B6 B6 B6 CD    DEFB   0D9H,0B6H,0B6H,0B6H,0CDH    ;S
3BF8  FE FE 80 FE FE    DEFB   0FEH,0FEH,80H,0FEH,0FEH     ;T
3BFD  C0 BF BF BF C0    DEFB   0C0H,0BFH,0BFH,0BFH,0C0H    ;U
3C02  F8 E7 9F E7 F8    DEFB   0F8H,0E7H,9FH,0E7H,0F8H     ;V
3C07  80 DF E7 DF 80    DEFB   80H,0DFH,0E7H,0DFH,80H      ;W
3C0C  9C ED F7 EB 9C    DEFB   9CH,0EDH,0F7H,0EBH,9CH      ;X
3C11  FC FB 87 FB FC    DEFB   0FCH,0FBH,87H,0FBH,0FCH     ;Y
3C16  9E AE B6 BA BC    DEFB   9EH,0AEH,0B6H,0BAH,0BCH     ;Z
3C1B  FF 80 BE BE FF    DEFB   0FFH,80H,0BEH,0BEH,0FFH     ;[
3C20  FD FB F7 EF DF    DEFB   0FDH,0FBH,0F7H,0EFH,0DFH    ;\
3C25  FF BE BE 80 FF    DEFB   0FFH,0BEH,0BEH,80H,0FFH     ;]
3C2A  FB FD 80 FD FB    DEFB   0FBH,0FDH,80H,0FDH,0FBH     ;^
3C2F  F7 E3 D6 F7 F7    DEFB   0F7H,0E3H,0D6H,0F7H,0F7H    ;_
3C34  FF FF FF FF FF    DEFB   0FFH,0FFH,0FFH,0FFH,0FFH    `
3C39  FF FF A0 FF FF    DEFB   0FFH,0FFH,0A0H,0FFH,0FFH    ;!
3C3E  FF F8 FF F8 FF    DEFB   0FFH,0F8H,0FFH,0F8H,0FFH    ;"
3C43  EB 80 EB 80 ED    DEFB   0EBH,80H,0EBH,80H,0EDH      ;#
3C48  DB D6 80 D6 ED    DEFB   0DBH,0D6H,80H,0D6H,0EDH     ;$
3C4D  D9 E9 F7 CB CD    DEFB   0D9H,0E9H,0F7H,0CBH,0CDH    ;%
3C52  C9 D6 A9 DF AF    DEFB   0C9H,0D6H,0A9H,0DFH,0AFH    ;&
3C57  F7 F8 FC FF FF    DEFB   0F7H,0F8H,0FCH,0FFH,0FFH    ;'
3C5C  FF E3 DD BE FF    DEFB   0FFH,0E3H,0DDH,0BEH,0FFH    ;(
3C61  FF BE DD E3 FF    DEFB   0FFH,0BEH,0DDH,0E3H,0FFH    ;)
3C66  D6 E3 80 E3 D5    DEFB   0D6H,0E3H,80H,0E3H,0D5H     ;*
3C6B  F7 F7 C1 F7 F7    DEFB   0F7H,0F7H,0C1H,0F7H,0F7H    ;+
3C70  DF C7 F7 FF FF    DEFB   0DFH,0C7H,0F7H,0FFH,0FFH    ,
3C75  F7 F7 F7 F7 F7    DEFB   0F7H,0F7H,0F7H,0F7H,0F7H    ;-
3C7A  FF 9F 9F FF FF    DEFB   0FFH,09FH,09FH,0FFH,0FFH    ;.
3C7F  DE EF F7 FB FD    DEFB   0DEH,0EFH,0F7H,0FBH,0FDH    ;/
```

```
3C84  C1 AE B6 BA C1    DEFB   0C1H,0AEH,0B6H,0BAH,0C1H      ;0
3C89  FF BD 80 BF FF    DEFB   0FFH,0BDH,80H,0BFH,0FFH       ;1
3C8E  9D AE B6 BA BD    DEFB   9DH,0AEH,0B6H,0BAH,0BDH       ;2
3C93  DD BB BB BB C9    DEFB   0DDH,0BBH,0BBH,0BBH,0C9H      ;3
3C98  E7 EB ED 80 EF    DEFB   0E7H,0EBH,0EDH,80H,0EFH       ;4
3C9D  D8 BA DA DA C6    DEFB   0D8H,0BAH,0DAH,0DAH,0C6H      ;5
3CA2  C1 B6 B6 B6 CF    DEFB   0C1H,0B6H,0B6H,0B6H,0CFH      ;6
3CA7  FC FE 86 FA FC    DEFB   0FCH,0FEH,86H,0FAH,0FCH       ;7
3CAC  C9 B6 B6 B6 C9    DEFB   0C9H,0B6H,0B6H,0B6H,0C9H      ;8
3CB1  F9 B6 B6 B6 C1    DEFB   0F9H,0B6H,0B6H,0B6H,0C1H      ;9
3CB6  FF C9 C9 FF FF    DEFB   0FFH,0C9H,0C9H,0FFH,0FFH      ;:
3CBB  BF C4 E4 FF FF    DEFB   0BFH,0C4H,0E4H,0FFH,0FFH      ;;
3CC0  F7 EB DD DE DE    DEFB   0F7H,0EBH,0DDH,0DEH,0DEH      ;<
3CC5  EB EB EB EB EB    DEFB   0EBH,0EBH,0EBH,0EBH,0EBH      ;=
3CCA  DE DE DD EB F7    DEFB   0DEH,0DEH,0DDH,0EBH,0F7H      ;>
3CCF  FD FE A6 FA FD    DEFB   0FDH,0FEH,0A6H,0FAH,0FDH      ;?

                        ************************************

                        Fehlermeldung ausgeben
                        Eing.: E = Fehlernummer
                               HL = Adresse der Fehler-Tabelle
3CD4  CB 3D             SRL    E               ;Fehlernummer / 2
3CD6  1C                INC    E               ;+ 1
3CD7  7E                LD     A,(HL)          ;Byte aus Fehler-Tabelle laden
3CD8  23                INC    HL              ;Tabellenadresse + 1
3CD9  B7                OR     A               ;neue Meldung ?
3CDA  F2 D7 3C          JP     P,3CD7H         ;nein!
3CDD  1D                DEC    E               ;Fehlernummer - 1 = 0 ?
3CDE  20 F7             JR     NZ,3CD7H        ;nein, nicht die richtige Meldung
3CE0  E6 7F             AND    7FH             ;Bit 7 löschen
3CE2  CD 2A 03          CALL   032AH           ;Byte ausgeben
3CE5  7E                LD     A,(HL)          ;nächstes Byte aus Fehlertab. laden
3CE6  23                INC    HL              ;Tabellenadresse + 1
3CE7  B7                OR     A               ;neue Meldung ?
3CE8  F2 E2 3C          JP     P,3CE2H         ;nein, Byte ausgeben
3CEB  C9                RET                    ;ja, fertig

                        ************************************

                        Tabelle der Fehlermeldungen

3CEC  CE                DEFB   'N'+80H         ;NEXT WITHOUT FOR
3CED  45 58 54 20 57    DEFM   'EXT WITHOUT FOR'
```

```
      49 54 48 4F 55
      54 20 46 4F 52
3CFC  D3              DEFB  'S'+80H          ;SYNTAX
3CFD  59 4E 54 41 58  DEFM  'YNTAX'
3D02  D2              DEFB  'R'+80H          ;RETURN WITHOUT GOSUB
3D03  45 54           DEFM  'ET'
3D05  27              DEFB  27H
3D06  4E 20 57 49 54  DEFM  'N WITHOUT GOSUB'
      48 4F 55 54 20
      47 4F 53 55 42
3D15  CF              DEFB  'O'+80H          ;OUT OF DATA
3D16  55 54 20 4F 46  DEFM  'UT OF DATA'
      20 44 41 54 41
3D20  C6              DEFB  'F'+80H          ;FUNCTION CODE
3D21  55 4E 43 54 49  DEFM  'UNCTION CODE'
      4F 4E 20 43 4F
      44 45
3D2D  CF              DEFB  'O'+80H          ;OVERFLOW
3D2E  56 45 52 46 4C  DEFM  'VERFLOW'
      4F 57
3D35  CF              DEFB  'O'+80H          ;OUT OF MEMORY
3D36  55 54 20 4F 46  DEFM  'UT OF MEMORY'
      20 4D 45 4D 4F
      52 59
3D42  D5              DEFB  'U'+80H          ;UNDEFINED STATEMENT
3D43  4E 44 45 46     DEFM  'NDEF'
3D47  27              DEFB  27H
3D48  44 20 53 54 41  DEFM  'D STATEMENT'
      54 45 4D 45 4E
      54
3D53  C2              DEFB  'B'+80H          ;BAD SUBSCRIPT
3D54  41 44 20 53 55  DEFM  'AD SUBSCRIPT'
      42 53 43 52 49
      50 54
3D60  D2              DEFB  'R'+80H          ;REDIMENSIONED ARRAY
3D61  45 44 49 4D     DEFM  'EDIM'
3D65  27              DEFB  27H
3D66  44 20 41 52 52  DEFM  'D ARRAY'
      41 59
3D6D  C4              DEFB  'D'+80H          ;DIVISION BY ZERO
3D6E  49 56 49 53 49  DEFM  'IVISION BY ZERO'
      4F 4E 20 42 59
      20 5A 45 52 4F
3D7D  C9              DEFB  'I'+80H          ;ILLEGAL DIRECT
```

```
3D7E  4C 4C 45 47 41     DEFM   'LLEGAL DIRECT'
      4C 20 44 49 52
      45 43 54
3D8B  D4                 DEFB   'T'+80H          ;TYPE MISMATCH
3D8C  59 50 45 20 4D     DEFM   'YPE MISMATCH'
      49 53 4D 41 54
      43 48
3D98  CF                 DEFB   'O'+80H          ;OUT OF SPACE
3D99  55 54 20 4F 46     DEFM   'UT OF SPACE'
      20 53 50 41 43
      45
3DA4  D3                 DEFB   'S'+80H          ;STRING TOO LONG
3DA5  54 52 49 4E 47     DEFM   'TRING TOO LONG'
      20 54 4F 4F 20
      4C 4F 4E 47
3DB3  C6                 DEFB   'F'+80H          ;FORMULA TOO COMPLEX
3DB4  4F 52 4D 55 4C     DEFM   'ORMULA TOO COMPLEX'
      41 20 54 4F 4F
      20 43 4F 4D 50
      4C 45 58
3DC6  C3                 DEFB   'C'+80H          ;CAN'T CONTINUE
3DC7  41 4E              DEFM   'AN'
3DC9  27                 DEFB   27H
3DCA  54 20 43 4F 4E     DEFM   'T CONT'
      54
3DD0  CE                 DEFB   'N'+80H          ;NO RESUME
3DD1  4F 20 52 45 53     DEFM   'O RESUME'
      55 4D 45
3DD9  D2                 DEFB   'R'+80H          ;RESUME WITHOUT ERROR
3DDA  45 53 55 4D 45     DEFM   'ESUME WITHOUT'
      20 57 49 54 48
      4F 55 54
3DE7  D5                 DEFB   'U'+80H          ;UNPRINTABLE ERROR
3DE8  4E 50 52 49 4E     DEFM   'NPRINTABLE'
      54 41 42 4C 45
3DF2  CD                 DEFB   'M'+80H          ;MISSING OPERAND
3DF3  49 53 53 49 4E     DEFM   'ISSING OPERAND'
      47 20 4F 50 45
      52 41 4E 44
3E01  C2                 DEFB   'B'+80H          ;BAD FILE DATA
3E02  41 44 20 46 49     DEFM   'AD FILE DATA'
      4C 45 20 44 41
      54 41
3E0E  C4                 DEFB   'D'+80H          ;DISK COMMAND
```

```
3E0F  49 53 4B 20 43    DEFM   'ISK COMMAND'
      4F 4D 4D 41 4E
      44
3E1A  3F 53 59 4E 54    DEFM   '?SYNTAX ERROR'
      41 58 20 45 52
      52 4F 52
3E27  0D 00             DEFB   0DH,00H

                        ************************************

                        wenn Puffer leer, ein Leerzeichen in Puffer schreiben
3E29  7E                LD     A,(HL)          ;Zeichen aus Puffer laden
3E2A  B7                OR     A               ;= 0 ?
3E2B  20 07             JR     NZ,3E34H        ;nein, nicht leer!
3E2D  3E 20             LD     A,' '           ;Leerzeichen laden
3E2F  77                LD     (HL),A          ;und an Pufferanfang
3E30  23                INC    HL              ;Endekennung hinter Leerzeichen
3E31  AF                XOR    A               ;(= X'00')
3E32  77                LD     (HL),A
3E33  2B                DEC    HL              ;Pufferzeiger wieder auf Anfang
3E34  2B                DEC    HL              ;Pufferzeiger auf Byte vor Puffer
3E35  F1                POP    AF              ;Kenner wieder laden
3E36  C9                RET                    ;fertig!

                        ************************************

                        Teil der Initialisierung
                        Default-Farbe = gelb setzen
3E37  32 7D 78          LD     (787DH),A       ;Interrupt-RAM-Ausgang = Return
3E3A  3E 10             LD     A,10H           ;Farbe gelb in Farbkenner
3E3C  32 46 78          LD     (7846H),A
3E3F  C9                RET

                        ************************************

                        Zusatzroutine zum Einlesen einer Zeile (RDLINE)
                        bei grünem Hintergrund und schwarzer Zeichendarstellung

                        Übertragen der Daten vom Bild zum I/O-Buffer
3E40  7E                LD     A,(HL)          ;Zeichen vom Bild laden
3E41  CB 77             BIT    6,A             ;Inverse-Zeichen ?
3E43  28 05             JR     Z,3E4AH         ;ja!
3E45  FE 80             CP     80H             ;Grafik-Zeichen ?
3E47  DA 5D 3E          JP     C,3E5DH         ;nein!
```

```
3E4A  C1           POP   BC          ;wenn nicht INPUT, dann sind
                                     ;Grafik und Inverse nur in
                                     ;Strings zugelassen
3E4B  11 53 3E     LD    DE,3E53H    ;Rücksprungadresse in Stack
3E4E  D5           PUSH  DE
3E4F  C5           PUSH  BC
3E50  C3 02 05     JP    0502H       ;Textendekennung prüfen (BREAK?)

3E53  D8           RET   C           ;BREAK! zurück zum BASIC
3E54  21 1A 3E     LD    HL,3E1AH    ;Text "SYNTAX ERROR" adressieren
3E57  CD A7 28     CALL  28A7H       ;und ausgeben
3E5A  C3 E3 03     JP    03E3H       ;zurück zur Zeileneingabe

3E5D  FE 62        CP    '"'+40H     ;Stringkennzeichnung '"' ?
3E5F  20 39        JR    NZ,3E9AH    ;nein, weiter
3E61  E6 BF        AND   0BFH        ;Bit 6 löschen
3E63  12           LD    (DE),A      ;Zeichen in I/O-Buffer
3E64  23           INC   HL          ;Bildadresse + 1
3E65  13           INC   DE          ;Bufferadresse + 1
3E66  05           DEC   B           ;Zeichenzähler - 1
3E67  CA EE 04     JP    Z,04EEH     ;wenn 0, übernahme beenden

                   Einsprung von 3EAF bei grünem Hintergrund
3E6A  7E           LD    A,(HL)      ;Zeichen aus Bild laden
3E6B  CB 7F        BIT   7,A         ;Grafik-Zeichen ?
3E6D  20 06        JR    NZ,3E75H    ;ja!
3E6F  CB 77        BIT   6,A         ;invertiertes Zeichen
3E71  20 0C        JR    NZ,3E7FH    ;nein!
3E73  18 06        JR    3E7BH       ;ja!
3E75  E6 8F        AND   8FH         ;Grafik, Bits 4,5,6 löschen
3E77  F6 80        OR    80H         ;Bit 7 setzen
3E79  18 17        JR    3E92H
3E7B  F6 C0        OR    0C0H        ;Bits 6 und 7 setzen
3E7D  18 13        JR    3E92H

3E7F  FE 62        CP    '"'+40H     ;ist es ein '"' ?
3E81  20 09        JR    NZ,3E8CH    ;nein!
3E83  E5           PUSH  HL          ;HL retten
3E84  21 39 78     LD    HL,7839H    ;Flag 2 adressieren
3E87  CB 66        BIT   4,(HL)      ;INPUT-Kommando ?
3E89  E1           POP   HL          ;HL wieder laden
3E8A  28 0E        JR    Z,3E9AH     ;nein - ab jetzt Grafik und
                                     ;Inverse nicht erlaubt
3E8C  CB 6F        BIT   5,A         ;Zeichen in echten ASCII-Code
```

```
3E8E  28 02     JR      Z,3E92H         ;umwandeln, wenn falsch
3E90  E6 BF     AND     0BFH            ;(Bit 6 ggf. löschen)
3E92  12        LD      (DE),A          ;Zeichen in I/O-Buffer
3E93  23        INC     HL              ;Bildadresse + 1
3E94  13        INC     DE              ;Bufferadresse + 1
3E95  10 D3     DJNZ    3E6AH           ;Zähler - 1
3E97  C3 EE 04  JP      04EEH           ;= 0, dann fertig

3E9A  CB 6F     BIT     5,A             ;Zeichen in echten ASCII-Code
3E9C  28 02     JR      Z,3EA0H         ;umwandeln, wenn falsch
3E9E  E6 BF     AND     0BFH            ;(Bit 6 ggf. löschen)
3EA0  12        LD      (DE),A          ;Zeichen in I/O-Buffer
3EA1  23        INC     HL              ;Bildadresse + 1
3EA2  13        INC     DE              ;Bufferadresse + 1
3EA3  10 9B     DJNZ    3E40H           ;Zähler - 1
3EA5  C3 EE 04  JP      04EEH           ;= 0, übertragung beendet

3EA8  3A 18 78  LD      A,(7818H)       ;abhängig vom Hintergrund in
3EAB  B7        OR      A               ;entsprechende Routine verzw.
3EAC  C2 B8 04  JP      NZ,04B8H        ;schwarzer Hintergrund
3EAF  C3 6A 3E  JP      3E6AH           ;grüner Hintergrund

                ************************************

                Zeichen invertieren
3EB2  3A 18 78  LD      A,(7818H)       ;Hintergrund-Flag laden
3EB5  B7        OR      A               ;schwarzer Hintergrund ?
3EB6  20 03     JR      NZ,3EBBH        ;ja!
3EB8  CB B6     RES     6,(HL)          ;grün, Bit 6 löschen
3EBA  C9        RET
3EBB  CB F6     SET     6,(HL)          ;schwarz, Bit 6 setzen
3EBD  C9        RET

                ************************************

                Löschzeichen für Bildlöschroutine bereitstellen
3EBE  3A 18 78  LD      A,(7818H)       ;Hintergrund-Flag laden
3EC1  B7        OR      A               ;schwarzer Hintergrund ?
3EC2  3E 20     LD      A,' '           ;Leerzeichen laden
3EC4  20 02     JR      NZ,3EC8H        ;schwarz, ist so ok!
3EC6  F6 40     OR      40H             ;grün, zus. Bit 6 setzen
3EC8  77        LD      (HL),A          ;Leerzeichen in Bild
3EC9  C9        RET
```

```
                        ************************************

                        bei Bildschirmausgabe Invertierung abhängig vom
                        Hintergrund ausführen
3ECA  F5        PUSH    AF              ;auszugebendes Zeichen sichern
3ECB  3A 18 78  LD      A,(7818H)       ;Hintergrund-Flag laden
3ECE  B7        OR      A               ;schwarzer Hintergrund ?
3ECF  28 07     JR      Z,3ED8H         ;nein - grün!
3ED1  F1        POP     AF              ;Zeichen wieder laden
3ED2  E6 3F     AND     3FH             ;Bits 6 und 7 löschen
3ED4  E5        PUSH    HL              ;Hauptroutine invertiert,
3ED5  C3 AB 31  JP      31ABH           ;falls erforderlich.

3ED8  F1        POP     AF              ;Zeichen wieder laden
3ED9  F6 40     OR      40H             ;Bit 6 setzen
3EDB  E5        PUSH    HL              ;HL sichern
3EDC  21 38 78  LD      HL,7838H        ;Flag 1 adressieren
3EDF  CB 4E     BIT     1,(HL)          ;invertieren?
3EE1  E1        POP     HL              ;HL wiederherstellen
3EE2  28 02     JR      Z,3EE6H         ;nein!
3EE4  E6 BF     AND     0BFH            ;Bit 6 löschen
3EE6  C3 B5 31  JP      31B5H           ;fertig

                        ************************************

                        Zeichen auf Leerzeichen (Blank) testen
3EE9  3A 18 78  LD      A,(7818H)       ;Hintergrund-Flag laden
3EEC  B7        OR      A               ;schwarzer Hintergrund ?
3EED  7E        LD      A,(HL)          ;Zeichen laden
3EEE  20 03     JR      NZ,3EF3H        ;ja, schwarz!
3EF0  FE 60     CP      ' '+40H         ;mit Bit 6 = 1 prüfen
3EF2  C9        RET
3EF3  FE 20     CP      ' '             ;mit Bit 6 = 0 prüfen
3EF5  C9        RET

                        ************************************

                        bei INSERT und RUBOUT Leerzeichen einfügen
3EF6  3A 18 78  LD      A,(7818H)       ;Hintergrund-Flag laden
3EF9  B7        OR      A               ;schwarz ?
3EFA  3E 20     LD      A,' '           ;Leerzeichen laden
3EFC  20 02     JR      NZ,3F00H        ;ja!
3EFE  F6 40     OR      40H             ;bei grün Bit 6 setzen
3F00  12        LD      (DE),A          ;in Bildspeicher
```

```
3F01  C9              RET

                      ************************************

                      für Roll-Routinen Zeilenlöschung vorbereiten
3F02  06 20           LD    B,32          ;Länge einer Zeile laden
3F04  3A 18 78        LD    A,(7818H)     ;Hintergrund-Flag laden
3F07  B7              OR    A             ;schwarz ?
3F08  3E 20           LD    A,' '         ;Leerzeichen laden
3F0A  C0              RET   NZ            ;ja, fertig
3F0B  F6 40           OR    40H           ;bei grün Bit 6 setzen
3F0D  C9              RET

                      ************************************

                      Hilfsroutine für das Laden von Kassette
                      zur korrekten Darstellung der Meldungen,
                      abhängig vom Hintergrund.

                      Aufruf von 3809H (allg. Meldungen in letzte Zeile)
3F0E  11 E0 71        LD    DE,71E0H      ;letzte Zeile adressieren
3F11  3A 18 78        LD    A,(7818H)     ;Hintergrund-Flag laden
3F14  B7              OR    A             ;schwarz ?
3F15  C0              RET   NZ            ;ja, keine Aktion
3F16  F1              POP   AF            ;Stack bereinigen
3F17  7E              LD    A,(HL)        ;Textzeichen laden
3F18  B7              OR    A             ;Textende ?
3F19  C8              RET   Z             ;ja, fertig
3F1A  CB B7           RES   6,A           ;Bit 6 löschen
3F1C  12              LD    (DE),A        ;Zeichen invertiert ausgeben
3F1D  13              INC   DE            ;Bildadresse + 1
3F1E  23              INC   HL            ;Textadresse + 1
3F1F  18 F6           JR    3F17H         ;nächstes Byte

                      Aufruf von 382BH (Ausgabe des Dateikenners)
3F21  3A 18 78        LD    A,(7818H)     ;Hintergrund-Flag laden
3F24  B7              OR    A             ;schwarz ?
3F25  7E              LD    A,(HL)        ;Kenner laden
3F26  20 07           JR    NZ,3F2FH      ;ja!
3F28  CB F7           SET   6,A           ;grün, Bit 6 setzen
3F2A  12              LD    (DE),A        ;ins Bild übertragen
3F2B  13              INC   DE            ;Bildadresse + 1
3F2C  3E 7A           LD    A,':'+40H     ;Trennzeichen f. grün laden
3F2E  C9              RET
```

```
3F2F  12          LD    (DE),A        ;Kenner ins Bild (f. schwarz)
3F30  3E 3A       LD    A,':'         ;Trennzeichen laden
3F32  C9          RET

                  Aufruf von 3837H (Ausgabe des Datei-/Programm-Namen)
3F33  F5          PUSH  AF            ;Zeichen sichern
3F34  3A 18 78    LD    A,(7818H)     ;Hintergrund-Flag laden
3F37  B7          OR    A             ;schwarz ?
3F38  20 05       JR    NZ,3F3FH      ;ja!
3F3A  F1          POP   AF            ;grün! Zeichen wieder laden
3F3B  F6 40       OR    40H           ;Bit 6 setzen (f. schwarze Darst.)
3F3D  12          LD    (DE),A        ;Zeichen auf Bild ausgeben
3F3E  C9          RET
3F3F  F1          POP   AF            ;schwarz! Zeichen wieder laden
3F40  E6 3F       AND   3FH           ;Bits 6 und 7 löschen
3F42  12          LD    (DE),A        ;Zeichen auf Bild ausgeben
3F43  C9          RET

                  ************************************

                  Hilfsroutine zur COPY-Anweisung
                  aufgerufen von 392DH
3F44  F5          PUSH  AF            ;Zeichen sichern
3F45  3A 18 78    LD    A,(7818H)     ;Hintergrund-Flag laden
3F48  B7          OR    A             ;schwarz ?
3F49  20 09       JR    NZ,3F54H      ;ja!
3F4B  F1          POP   AF            ;grün! Zeichen wieder laden
3F4C  CB 77       BIT   6,A           ;invertiertes Zeichen ?
3F4E  C2 38 39    JP    NZ,3938H      ;nein, normale Ausgabe
3F51  C3 31 39    JP    3931H         ;ja, invertierte Ausgabe

3F54  F1          POP   AF            ;schwarz! Zeichen wieder laden
3F55  CB 77       BIT   6,A           ;invertiertes Zeichen ?
3F57  CA 38 39    JP    Z,3938H       ;nein, normale Ausgabe
3F5A  C3 31 39    JP    3931H         ;ja, invertierte Ausgabe
3F5D  C3 31 39    JP    3931H         ;nicht benutzt!

                  ************************************

                  Hilfsroutine zur Zeichenausgabe auf dem Bildschirm
                  Anpassung der Invertierung an die Hintergrundfarbe
                  Aufruf von 3149H
3F60  F5          PUSH  AF            ;Zeichen sichern
3F61  3A 18 78    LD    A,(7818H)     ;Hintergrund-Flag laden
```

```
3F64  B7          OR    A             ;schwarz ?
3F65  20 06       JR    NZ,3F6DH      ;ja!
3F67  F1          POP   AF            ;grün! Zeichen wieder laden
3F68  E6 3F       AND   3FH           ;Bits 6 und 7 löschen
3F6A  C3 54 31    JP    3154H
3F6D  F1          POP   AF            ;schwarz! Zeichen wieder laden
3F6E  E6 7F       AND   7FH           ;Bit 7 löschen
3F70  C3 54 31    JP    3154H

                  ************************************

                  Hilfsroutine zum Einlesen von Kassette
                  Aufruf von 369CH
3F73  CD 75 37    CALL  3775H         ;Byte von Kassette lesen
3F76  D0          RET   NC            ;ok!
3F77  E1          POP   HL            ;Rücksprungadresse vom Stack
3F78  C3 11 37    JP    3711H         ;Lesefehler!

                  ************************************

                  Bei Wahl einer neuen Hintergrundfarbe
                  Bildschirm-Inhalt umwandeln
                  Aufruf erfolgt von der Interrupt-Service-Routine
3F7B  3A 19 78    LD    A,(7819H)     ;akt. Hintergrund-Flag
3F7E  47          LD    B,A
3F7F  3A 18 78    LD    A,(7818H)     ;= gewähltem Hintergrund ?
3F82  B8          CP    B
3F83  CA E8 30    JP    Z,30E8H       ;ja, zur Zeichenausgabe
3F86  32 19 78    LD    (7819H),A     ;akt. Hintergrund-Flag aktualis.
3F89  21 00 70    LD    HL,7000H      ;Bildanfangsadresse laden
3F8C  01 00 02    LD    BC,512        ;Bildschirmgröße
3F8F  7E          LD    A,(HL)        ;Zeichen laden
3F90  B7          OR    A             ;Blockgrafik ?
3F91  FA 97 3F    JP    M,3F97H       ;ja, unverändert lassen
3F94  EE 40       XOR   40H           ;Bit 6 (Invertierung) kippen
3F96  77          LD    (HL),A        ;und zurückschreiben
3F97  23          INC   HL            ;Bildadresse + 1
3F98  0B          DEC   BC            ;Zeichenzähler - 1
3F99  78          LD    A,B           ;= 0 ?
3F9A  B1          OR    C
3F9B  20 F2       JR    NZ,3F8FH      ;nein, nächstes Zeichen
3F9D  C3 E8 30    JP    30E8H         ;ja, fertig!

                  ************************************
```

```
                        Prüfen, ob während der Initialisierung
                        die CTRL-Taste betätigt ist
3FA0  3A FD 68          LD      A,(68FDH)       ;Tastaturzeile 2 laden
3FA3  CB 57             BIT     2,A             ;CTRL-Taste betätigt ?
3FA5  3E 20             LD      A,' '           ;Leerzeichen in A
3FA7  20 08             JR      NZ,3FB1H        ;nicht betätigt!
3FA9  F6 40             OR      40H             ;Bit 6 setzen (schwarzer Hintergr.)
3FAB  32 18 78          LD      (7818H),A       ;Hintergrund-Flags für schwarz
3FAE  32 19 78          LD      (7819H),A       ;setzen
3FB1  32 3C 78          LD      (783CH),A       ;Leerzeichen als Cursor-Sicherung
3FB4  C3 C9 01          JP      01C9H           ;Fortsetzung der Initialisierung
```

Verzeichnis der RAM-Variablen , -Vektoren und -Zeiger, die vom
BASIC - Interpreter angesprochen werden. (mit Querverweisen zum ROM)

Der Bereich 7800H bis 7835H wird bei der System-Initialisierung
aus dem ROM-Bereich vorbelegt.

```
7800  C3 96 1C        JP     1C96H        ;RST 8 - Vektor
                                          ;0008, 0679
7803  C3 78 1D        JP     1D78H        ;RST 10 - Vektor
                                          ;0010
7806  C3 90 1C        JP     1C90H        ;RST 18 - Vektor
                                          ;0018
7809  C3 D9 25        JP     25D9H        ;RST 20 - Vektor
                                          ;0020
780C  C9 00 00        RET                 ;RST 28 - Vektor
                                          ;0028
780F  C9 00 00        RET                 ;RST 30 - Vektor
                                          ;0030
7812  FB              EI                  ;RST 38 - Vektor
7813  C9 00           RET                 ;(wird nicht angesprochen)

                      Tastatur Device-Control-Block (DCB)
7815  01              DCB-Kenner          ;002B
7816  F4 2E           Treiber-Adresse
7818  00              Hintergrund-Flag    ;048E 3EA8 3EB2 3EBE 3ECB 3EE9 3EF6
                      (0=grün, 1=schwarz) ;3F04 3F11 3F21 3F34 3F45 3F61 3F7F
                                          ;3FAB
7819  00               akt. Hintergrund   ;3F7B 3F86 3FAE
781A  00
781B  4B 49           'KI'

                      Bildschirm Device-Control-Block (DCB)
                      (im LASER 110-310 unbenutzt)
781D  00              DCB-Kenner (gelöscht) ;0033

781E  00 00           Zeiger auf Programm-Anfangsadresse
                      bei CLOAD           ;368D 36CB

7820  00 70           Cursor-Adresse      ;0050 0311 034D 03E8 0419 041E 042A
                                          ;0468 0505 055B 2083 2EEC 3033 3111
                                          ;3129 31F5 3202 3207 323B 3247 325C
                                          ;3266 3276 3280 328A 3295 32B4 32C2
                                          ;32D7 3308 332C 33A3 33D0 35CB 3657
```

```
                                        ;3723

7822  00
7823  00 00             Prüfsumme bei Kassetten Ein-/Ausgabe
                                        ;34BE 367D

                        Drucker Device-Control-Block (DCB)
7825  06                DCB-Kenner        ;003B
7826  8D 05             Treiber-Adresse
7828  43                Zeilen/Seite+1
7829  00                Zeilenzähler
782A  00
782B  50 52             'PR'

782D  C3 00 50          JP      5000H     ;unbenutzt
7830  C7 00 00          RST     0         ;unbenutzt

7833  3E 00             LD      A,0       ;bei unbekannter DCB-Kennung A=0
7835  C9                RET               ;03D1

7836                    Puffer B1 für 1. Tastencode bei gleichzeitig mehrfacher
                        Tastenbetätigung  ;0603 060B 0635 065D 066B 2F24 2FFB

7837                    Puffer B2 für 2. Tastencode bei gleichzeitig mehrfacher
                        Tastenbetätigung  ;05FF 062E 063B 0664 0668 2FFF

7838                    FLAG 1            ;0517 051C 05D7 05F4 0625 0656 2F0E
                        Bit 7 - CONTROL-Flag  ;2F20 2F83 2F9B 2FA9 2FCD 2FD7 2FF0
                        Bit 6 - REPEAT-Flag   ;3039 31AB 3EDC
                        Bit 5 - WAIT-Flag
                        Bit 4 - B2-Status-Flag
                        Bit 3 - B1-Status-Flag
                        Bit 2 - FUNCTION-Flag
                        Bit 1 - INVERSE-Flag
                        Bit 0 - SHIFT-Flag

7839                    FLAG 2            ;0183 03E3 0405 0425 04CE 051F 052C
                        Bit 7 - unbenutzt ;0567 2EC9 2EDC 3028 30AB 31D2 3212
                        Bit 6 - CRUN-Flag ;3253 326D 3430 3694 36D9 36E3 372F
                        Bit 5 - Ini-Flag f. gepufferte Ausgabe   ;3739 3758 3AFE
                        Bit 4 - Flag f. INPUT-Anweisung          ;3803 3E84
                        Bit 3 - VERIFY-Flag
                        Bit 2 - BREAK-Flag
                        Bit 1 - BUZZER-Flag
```

```
                    Bit 0 - Carriage-Return Flag

783A                Zeitzähler              ;05DF 05E6 2F15 2FC1 2FDE 2FE2 2FF4
                                            ;300A 300E 3012 3491

783B                INPUT/OUTPUT-Latch      ;2C41 2E73 2E78 2E7F 2E84 308C 3095
                                            ;345C 3489 352B 3542 35B6 35BF 38CA
                                            ;38CF 38DB 38E0 3914

783C                Zeichensicherung für Cursor-Darstellung
                                            ;0054 030E 32F5 3FB1

783D-7840           unbenutzt               ;0348

7841                Blink-Zähler            ;2EE2 2EE9 3030 3496

7842-7843           Zwischenspeicher bei Tastaturabfrage
                    (Zeile/Spalte)          ;0611 0643 2F6F

7844-7845           Zwischenspeicher bei Tastaturabfrage
                    (Matrix-Adresse)        ;0615 0647 2F73

7846                Farb-Code               ;0173 3150 38B9 3E3C

7847-784B           unbenutzt

784C                Ausgabe-Flag f. Meldungsausgabe bei Kassetten I/O
                    (>0 - Meldungen werden unterdrückt)
                                            ;35B1 3719 37F4 3804 3814

784D-787C           unbenutzt

787D  C9 00 00      RAM-Erweiterungsausgang der Interrupt-Service-Routine
                                            ;2EBC 3E37
```

```
                        Der Bereich 7880 - 78A5 wird bei der Initialisierung aus
                        dem ROM-Bereich gefüllt

                        Unterprogramm für Division
7880  D6 00            SUB   0             ;Subtraktion Z2 - Z1
                                            ;wird vor jedem Aufruf modifiziert.
                                            ;0075 08BB 08CA
7882  6F              LD    L,A
7883  7C              LD    A,H
7884  DE 00           SBC   A,0             ;08B6
7886  67              LD    H,A
7887  78              LD    A,B
7888  DE 00           SBC   A,0             ;08B1
788A  47              LD    B,A
788B  3E 00           LD    A,0             ;08C4 08D2 08F0 08F4
788D  C9              RET

                        USR - Startadresse
                        vorbesetzt mit FUNCTION-CODE Error
788E  4A 1E                                 ;2B15

7890  40 E6 4D          Multiplikator f. RND  ;14F0

                        Unterprogramm für INP
7893  DB 00            IN    A,(0)           ;2AF5 2AF2 2B11
7895  C9              RET

                        Unterprogramm für OUT
7896  D3 00            OUT   (0),A           ;2AFE 2B14

7899  00                INKEY$-Zwischenspeicher
                                            ;019F 01AD 1DA5
789A  00                letzter Fehlercode für ERR
                                            ;19B7 1A2B 1F8E 1FB8 24D0
789B  00                Druckerposition in der Zeile
                                            ;038F 03B1 03B7 20D5 211B 214E
789C  00                Geräte-Flag (0=Bildsch., 1=Drucker, 80=Kassette)
                                            ;032F 038C 2069 2098 20CC 2144 2169
                                            ;2171 2B2B 3B0C
789D  40                Zeilenlänge auf dem Bildschirm (vorbesetzt mit 64)
                                            ;20DD
789E  30                letzte Tabulator-Position (vorbesetzt mit 48)
                                            ;2123
789F  00                unbenutzt
```

```
78A0  47 7B     Anfangsadresse des String-Bereichs
                                ;00F6 1917 1B90 1E9C 1F4C 27E5 28C3
                                ;28F0
78A2  FE FF     aktuelle Zeilennummer  ;197E 1994 19A2 1A36 1CC1 1D41 1DC1
                                ;1DF2 1EB9 1EC9 1EF0 1FD6 231C 2829
                                ;2B36 36F8
78A4  E9 7A     Anfangsadresse des Programmtextes
                                ;191B 1AF8 1B2C 1B4D 1B5D 1D92 1F46
                                ;31DE 34C2 36D5
78A6-78A7       Spaltenzeiger für Ausgabebild
                                ;0410 0415 0551 2089 20E1 2153 27F5
                                ;30CE 3114 311A 31BF 31F2 3227 3235
                                ;3235 324F 328E 32A5 32B7 32BD 32CD
                                ;3302 3318 333A 3356 335F 33A8 33AE
                                ;33D6 341D 35D0 3728 3B21
78A7-78A8       Zeiger auf Ein-/Ausgabe-Puffer (ab 79E8)
                                ;008B 1ADB 1BC6 1C84 21AF 21C3 2B6A
                                ;2B7F 2E5C
78A9            Eingabe-Flag (0 = Kassette)
                                ;2186 21A3 220E

78AA            letzte Zufallszahl     ;150B 1526
78AB                                   ;01D5
78AC                                   ;1510 152F
78AD

78AE            Flag für DIM-Anweisung
                                ;260E 26EA 2707 272F 2757

78AF            Typ des Wertes im X-Register
                02 = Integer           ;01C4 09D3 0FE1 22FC 2374 2399 2410
                03 = String            ;2465 2509 2653 2716 27AF 27CB 280A
                04 = einfache Genauigkeit  ;2B91
                08 = doppelte Genauigkeit

78B0            Flag für Zwischencode-Erzeugung bei DATA
                Operationscode bei der Ausdrucksanalyse
                                ;1B1C 1BDC 1C67 2408 2451

78B1-78B2       Endadresse des BASIC-Speicherbereichs
                                ;00F2 0A9F 1B7A 1E84 28E6

78B3-78B4       Zeiger auf String-Zwischenspeicher
```

```
                                ;1B9E 1DBA 2889 289B 28F8 29F5 29FF

78B5-78D2       String-Zwischenspeicher (10 x 3 Bytes)
                (1 Byte - Länge, 2 Byte - Adresse im Stringbereich)
                                ;1B9B 1DB7 28F4

78D3-78D5       vorl. String-Zwischenspeicher
                (wie oben)      ;285A 2884 01B5 29B4 2A27 2A57

78D6-78D7       Zeiger auf letztes freies Byte im Stringbereich
                                ;1B7D 27E9 2897 28C7 28D3 28E9 2955
                                ;297F 29E8

78D8-78D9       Allg. Adresszwischenspeicher
                Formatflag f. Stringausgabe einer Zahl
                                ;0FDC 0FFB 1034 1289 235E 2368 23CE
                                ;2752 2784 2935 2940

78DA-78DB       DATA - Zeilennummer     ;1991 22AA

78DC            Flag zur Sperrung der Indizierung
                                ;1BAA 1CA3 2657 2665

78DD            RESUME/RETURN - Flag    ;1AAA 1EF8 1FEA

78DE            Zwischenpuffer für PRINT USING
                DATA-Flag für INPUT
                u.a.            ;217F 21F5 2207 226F 2CCB 2CD5 2DD9

78DF-78E0       allgemeiner Adress-Speicher
                z.B. Programmfortführung bei NEW
                Laufvariable bei FOR/NEXT
                Adr. d. Variablentabelle bei LET
                                ;1B61 1BAF 1D16 1F27 22BC 2328

78E1            AUTO-Eingabe - Flag (0 - kein AUTO)
                                ;1A5B 1B53

78E2-78E3       AUTO - Zeilennummer     ;1A3F 1A6E 2032
78E4-78E5       AUTO - Erhöhungswert    ;1A60 2019

78E6-78E7       Adresse der aktuellen Zeile (FFFF = Direktkommando)
                                ;19BA 1A9E 1D25 1DB4 1DCE 2196
```

78E8-78E9 Zeiger auf den BASIC-Stack
;19AE 1B95 1CB2 1D28 1EE5 22C6 2325

78EA-78EB Nummer der Zeile, in der der letzte Fehler auftrat
;19A5 19C1 1A02 1FD3 24DF

78EC-78ED Nummer der Zeile, in der der letzte Fehler auftrat
(.-Option bei LIST)
;19A8 1AB1 1E53 2B5B

78EE-78EF Adresse der Zeile, in der der Fehler auftrat
;19BD 1FCF

78F0-78F1 Adresse einer Fehlerbehandlungs-Routine (ON ERROR)
;19D0 1B74 1F84

78F2 Fehler - Flag (Fehler=255, RESUME=0)
;1986 19D6 1B6F 1FAF

78F3-78F4 Adresse des Dezimal-Punktes im Druck-Puffer
;10CE 1197 1296 2343 2346 2537 2703
;27C5

78F5-78F6 Zeilennummer, bei der die letzte Unterbrechung
stattfand (END, STOP, BREAK)
;19C9 1DCB 1DEF

78F7-78F8 Adresse der Zeile, in der die letzte Unterbrechung
stattfand ;19CD 1B77 1DE4

78F9-78FA Programm-Endadresse
Anfang der Variablen-Tabelle
;1AC2 1ACD 1B5A 1B83 1E90 1F53 266A
;2903 2BE5 2BF1 34D8 36BB

78FB-78FC Endadresse der Variablen-Tabelle
Anfang der Matrix-Tabelle
;1B86 266E 26BB 2907

78FD-78FE Anfangsadresse des freien Speichers
(hinter der Matrix-Tabelle)
;1964 1B89 26AB 26B6 2711 2779 27D4
;2922

```
78FF-7900               Zeiger auf DAT-Zeile   ;1D96 21F0


                        Typcode - Tabelle
7901                    A                      ;1B66 1E2A 264A
7902                    B
7903                    C
7904                    D
7905                    E
7906                    F
7907                    G
7908                    H
7909                    I
790A                    J
790B                    K
790C                    L
790D                    M
790E                    N
790F                    O
7910                    P
7911                    Q
7912                    R
7913                    S
7914                    T
7915                    U
7916                    V
7917                    W
7918                    X
7919                    Y
791A                    Z

791B                    TRACE -FLAG  (0 = TRON, AF = TROFF)
                                       ;1D44 1DF9


                        % - Register
791C  zus. Byte für rechts schieben            ;0B98 0CB6 0CE0 0CF7 0D0E 0D5A

      INT    STRING    SINGLE    DOUBLE
      ---    ------    ------    ------
791D                               LSB         ;0A08 0AE6 0BA0 0D20 0D36 0D48 0E1E
                                               ;12E2 23B7 2443
791E                               LSB
791F                               LSB         ;0AE9 243F
7920                               LSB         ;0AC5
```

7921	LSB	ADR	LSB	LSB	;01BF 073C 0866 099B 09A5 09B5 09BF ;09CB 0A03 0A80 0A9A 0ACC 0B46 0C5B ;0F37 0F4E 1343 1352 1426 1F3E 20C9 ;2395 23FD 2433 246D 248B 2545 2562 ;25D2 26DB 26E4 288C 2991 299E 29DA ;2CC6 2E2A 2E37
7922	MSB	ADR	LSB	LSB	
7923			MSB	MSB	;095A 0982 09AA 09BA 09DF 0A1A 0A62 ;0B8A 0CDA 0D90 0DCC 0E14 1422 2487
7924			EXP	EXP	;0719 0779 078E 0797 0810 08FD 0919 ;094D 0955 0969 0A8F 0B40 0B59 0C80 ;0D05 0E30 1208 1445 15C6 26D5

7925 Zwischenspeicher für Arithmetik-Operationen.
z.B. Vorzeichen ;07C3 0D15 1535

Y - Register
(Aufteilung wie X-Register)

7926-792E ;09F4 09FC 0A49 0D33 0D45 0DDF 0DFC
;0E0A 1213

792F unbenutzt ;10CA

7930-7949 Druck-Puffer ;0FF5 1037 1096

794A-7951 Zusätzliches Register für Multiplikationen und
Divisionen mit doppelter Genauigkeit
;0DF9 0E07 0E26

RAM-Vektoren für Disketten-Befehle
vorbesetzt mit 'JP 012DH' (DISK-COMMAND - Error)

7952	CVI-Anweisung	;0093 1626
7955	FN - Anweisung	;2524
7958	CVS-Anweisung	;1628
795B	DEF-Anweisung	;1882
795E	CVD-Anweisung	;162A
7961	EOF-Anweisung	;162C

```
7964    LOC-Anweisung          ;162E
7967    LOF-Anweisung          ;1630
796A    MKI$-Anweisung         ;1632
796D    MKS$-Anweisung         ;1634
7970    MKD$-Anweisung         ;1636
7973    CMD-Anweisung          ;182C
7976    TIME$-Anweisung        ;2510
7979    OPEN-Anweisung         ;1866
797C    FIELD-Anweisung        ;1868
797F    GET-Anweisung          ;186A
7982    PUT-Anweisung          ;186C
7985    CLOSE-Anweisung        ;186E
7988    LOAD-Anweisung         ;1870
798B    MERGE-Anweisung        ;1872
798E    NAME-Anweisung         ;1874
7991    KILL-Anweisung         ;1876
7994    & - Anweisung          ;24C8
7997    LSET-Anweisung         ;1878
799A    RSET-Anweisung         ;187A
799D    INSTR-Anweisung        ;2506
79A0    SAVE-Anweisung         ;187C
79A3    LINE-Anweisung         ;keine Referenz
```

RAM-Erweiterungsausgänge
vorbelegt mit 'RET'

79A6	aus ERROR-Routine	;19EC
79A9	aus USR-Routine	;27FE
79AC	Anfang BASIC-Schleife	;1A1C
79AF-79B1	unbenutzt	
79B2	aus Programm-Eingabe	;1AA1
79B5	Ende Programmeingabe	;1AEC 36EC
79B8	Ende Programmeingabe	;1AF2 36F2
79BB	aus NEW und END	;1B8C 1D80
79BE	Endabfrage PRINT	;2174
79C1	Datenausgabe	;032C
79C4	Einlesen v. Tastatur	;0358
79C7	RUN-Ausführung	;1EA6
79CA	Anfang PRINT-Anweisung	;206F
79CD	PRINT-Anweisung	;20C6
79D0	PRINT-Anweisung	;2103
79D3	PRINT-Anweisung	;2108 2141
79D6	INPUT-Anweisung	;219A
79D9	MID$ als Anweisung	;2AEC
79DC	INPUT-Anweisung	;2220
79DF	READ + INPUT + LIST	;2278 2B44

79E2-79E4 unbenutzt

79E5 3A 00 I/O-Buffer-Vorspann ;0080
79E7 2C ;1A73 370A

79E8-7A9C Ein/Ausgabe-Puffer ;0421 046B 0531 0542 0560 36FB

79F8 BASIC-Stack während der Initialisierung ;00AC

7A9D-7AAD Programm-/Dateiname - Zwischenspeicher bei Kassetten-Ein/Ausgabe
;3581 358E 3647

7AAE Spaltenanzeige auf dem Bildschirm
;2127 30D2

Zusätzlicher Ausgabepuffer für gepufferte Bildschirmausgabe

7AAF Anzahl Zeichen im Puffer
;053A 30B9 30C9 30E8 3102 349A 3B13
;3B1C

7AB0-7AB1 Puffer-Zeiger ;30C1 30C6 30FE 34A0

7AB2-7AD1 Puffer-Bereich ;30EE 349D 3626 363E 3644

7AD2 4 Byte-Puffer für Grafik-Druck, SOUND u. Kassette I/O
;05CD 2BFD 2C12 3623 3670 36C4 381D
;399A 3B7F
7AD3 ;3782 3788 39ED
7AD4
7AD5 ;3A02

7AD6 Zähler f. o.a.Puffer + Länge Namen bei Kassetten I/O
;357D 35AD 398F 3992 3A2D 3A36 3A59
;3A8E 3A96

Zeilenstatus für Bildschirm-Zeilen
(80=Einzelzeile, 81=Doppelzeile, 00=Folgezeile)

```
7AD7    Zeile 1                 ;03FB 32AC 333E 33C5 3406 3424
7AD8    Zeile 2
7AD9    Zeile 3
7ADA    Zeile 4
7ADB    Zeile 5
7ADC    Zeile 6
7ADD    Zeile 7
7ADE    Zeile 8
7ADF    Zeile 9
7AE0    Zeile 10
7AE1    Zeile 11
7AE2    Zeile 12
7AE3    Zeile 13
7AE4    Zeile 14
7AE5    Zeile 15                ;35D3 35DA
7AE6    Zeile 16                ;338E 339D 35DD

7AE7

7AE8    Programm-Anfang         ;00A8
```

Die Einsteiger-Modelle für Schüler und Studenten

LASER™
HOME-COMPUTER

LASER 210, 8 KByte RAM,
erweiterbar um 16 oder 64 KByte,
8 Farben, Programmsprache BASIC.

LASER 310 mit gleicher Ausstattung wie Laser 210,
aber 18 KByte RAM und mit Schreibmaschinen-Tastatur.

Floppy Disk Controller für 2 Laufwerke
mit LASER-DOS, Speicherkapazität 80 KByte.

Generalimporteur: SANYO VIDEO Vertrieb GmbH & Co.
Lange Reihe 29, D-2000 Hamburg 1, Tel. 040/28010 45-49

ENTSÄUERT
PAL 10/2016

VOGEL-BUCHVERLAG WÜRZBURG

Diese knapp gehaltene Einführung in die Welt der Home-Computer zeigt allen Einsteigern, welche Möglichkeiten der Home-Computer bieten kann. Der Autor veranschaulicht in bewährter Art und Weise, abgespeckt von überflüssigem Ballast, wo die Einsatzgebiete — die Stärken und die Schwächen dieser neuen Computer-Generation liegen. Typische Beispiele verdeutlichen die Problematik des Themas.

Sacht, Hans-J.
Home-Computer kurz und bündig
Reihe HC –
Mein Home-Computer
152 Seiten,
zahlr. Abbildungen,
20,– DM
ISBN 3-8023-**0790**-9

Tatzl, Gerfried
Die besten Anwendungen für Home-Computer
Reihe HC –
Mein Home-Computer
192 Seiten,
zahlr. Abbildungen,
30,– DM
ISBN 3-8023-**0787**-9

Das Hauptaugenmerk wird in diesem Buch nach einer kurzen Einführung auf Anwendungen gelegt, für die Home-Computer sinnvoll eingesetzt werden können. Besondere Bedeutung erhält dabei die Lösung von Aufgaben. Neben einigen Grafikanwendungen und Computerspielen werden Beispiele für Hobby, Haushalt, Textverarbeitung, Technik und Produktion gebracht.

Dieser BASIC-Sprachführer für Umsteiger bietet Hilfestellung beim Übergang von der Benutzung tastenprogrammierbarer Rechner unterschiedlicher Rechenlogik zu Geräten, die mit BASIC arbeiten. Der Benutzer lernt gleichzeitig das Übersetzen von Programmen aus einem Sprachsystem in ein anderes. Aber auch alle BASIC-Computer-Besitzer werden auf ihre Kosten kommen.

Tatzl, Gerfried
Vom Taschenrechner zum Home-Computer
Reihe HC –
Mein Home-Computer
272 Seiten,
zahlr. Abbildungen,
38,– DM
ISBN 3-8023-**0772**-0

Tatzl, Gerfried
Praktische Problemanalyse
Problem-Engineering
Reihe CHIP WISSEN
320 Seiten,
53 Abbildungen,
45,– DM
ISBN 3-8023-**0745**-3

Dieses Buch leistet einen Beitrag zur Behebung der vielzitierten Softwarekrise. Ohne den Leser in ein enges Denkschema zu pressen, wird vorwiegend die kreative und intuitive Seite angesprochen. Anhand verschiedener Beispiele wird in diesem Buch der Weg eines „springenden Funkens" verfolgt. So lassen sich optimale Problemgestaltungen ableiten und in entsprechende Programme umsetzen.

Schneller erfolgreich durch Computer-Bücher

Mein Home-Computer

Das Magazin für mehr Spaß beim Computern

Monat für Monat über 30 Seiten Programme

Und das bringt Mein Home-Computer jeden Monat:

- ✻ Programme für alle gängigen Home-Computer
- ✻ Anwendungsbeispiele aus der Praxis
- ✻ Marktübersicht, Tests und Kaufberatung für Zusatzgeräte und Home-Computer
- ✻ Schnellkurse für Einsteiger zum Sammeln
- ✻ Tips und Tricks
- ✻ Interessantes, Aktuelles und Unterhaltsames aus der Home-Computer-Szene
- ✻ News, Clubnachrichten

Holen Sie sich die neueste Ausgabe bei Ihrem Zeitschriftenhändler oder fordern Sie ein Kennenlernheft direkt beim Vogel-Verlag, Leserservice HC, Postfach 67 40, 8700 Würzburg, an.

In kurzer Zeit haben sich die Home-Computer Sanyo Laser und VZ 200 eine beachtliche Anhängerschaft erworben – vor allem jugendliche Fans. Ihr Spieltrieb ist ungebrochen, ebenso ihre Neugier und ihr Spaß am Experimentieren. Dazu gehört, daß man voll erkennt, was das Gerät zu bieten hat, daß man weiß, wie es «da drinnen» aussieht. Dort einzudringen und alles zu erforschen, helfen diese ROM-Listings: ein toller «Fahrplan» in die letzten Geheimnisse des ROM-Speichers, klar gegliedert und ausführlich kommentiert.

VOGEL-BUCHVERLAG
WÜRZBURG

ISBN 3-8023-0852-2